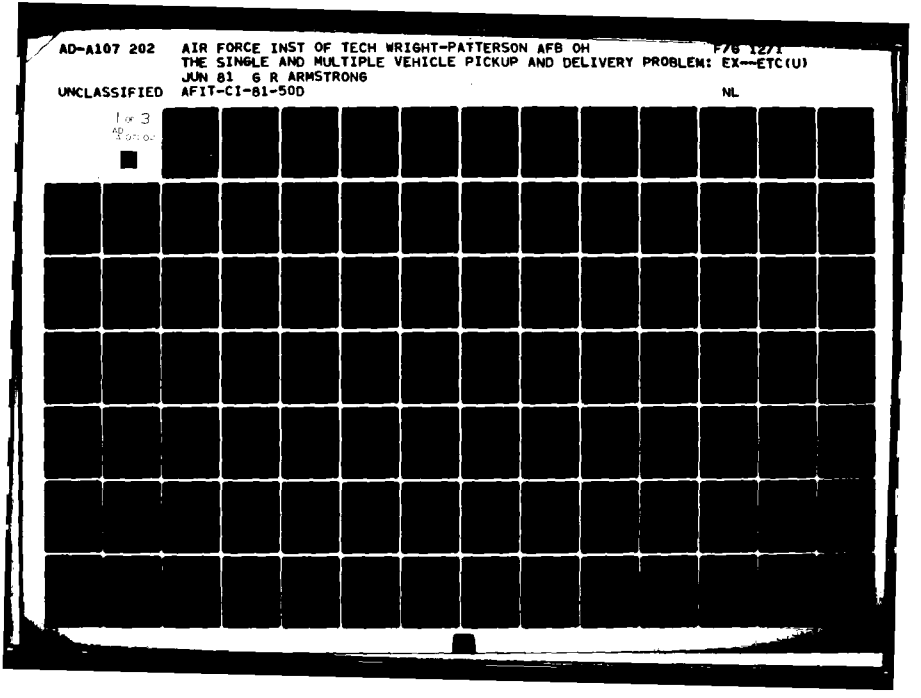
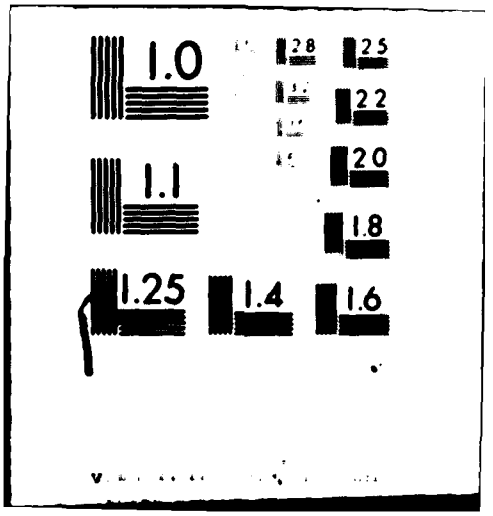
AD-A107 202 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH F/G 12/1
THE SINGLE AND MULTIPLE VEHICLE PICKUP AND DELIVERY PROBLEM: EX--ETC(U)
JUN 81 G R ARMSTRONG
UNCLASSIFIED AFIT-CI-81-50D NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1 REPORT NUMBER: 81-50D	2 GOVT ACCESSION NO.: AD-A107 202
3 RECIPIENT'S CATALOG NUMBER:	
4 TITLE (and Subtitle): The Single and Multiple Vehicle Pickup and Delivery Problem: Exact and Heuristic Algorithms.	5 TYPE OF REPORT & PERIOD COVERED: THESIS/DISSERTATION
	6 PERFORMING ORG. REPORT NUMBER:
7 AUTHOR(s): Gerald R. Armstrong	8 CONTRACT OR GRANT NUMBER(s):
9 PERFORMING ORGANIZATION NAME AND ADDRESS: AFIT STUDENT AT: Univ of Tennessee	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11 CONTROLLING OFFICE NAME AND ADDRESS: AFIT/NR WPAFB OH 45433	12 REPORT DATE: June 1981
	13 NUMBER OF PAGES: 185
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): LEVEL II	15 SECURITY CLASS (of this report): UNCLASS
	15a DECLASSIFICATION DOWNGRADING SCHEDULE:

16 DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 OCT 1981

FREDRIC C. LYNCH, Major, USAF
Director of Public Affairs
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

18 SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

Director of Public Affairs

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

81 10 27 256

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A107202

DTIC FILE COPY

ABSTRACT

The pickup and delivery problem (PUDP) represents a class of sequencing or routing problems where the key facet of the routing is that a pickup must precede the corresponding, subsequent delivery. Other considerations such as service time windows, quality of service parameters or operational constraints on either the driver or the vehicle are possible. As such, the PUDP is a constrained version of the ubiquitous travelling salesman problem (TSP), which seeks a minimum cost route that from an initial point visits each city or stop once and only once, ending at the initial stop. There are also similarities between the PUDP and the much studied vehicle routing problem (VRP), although the two problems are distinctly different because of the origin preceding destination requirement.

The TSP and VRP literature is extensive, offering both theory and algorithms for the solution of these problems. Given the similarities of these problems to the PUDP, those algorithms that performed well on TSP's and VRP's are discussed in detail and served as the basis for developing both exact and heuristic algorithms to solve the PUDP.

Assuming that all problem constraints are expressable in terms of stop numbers along a vehicle's route, dynamic programming can be used to optimally solve the problem. The algorithm developed is significantly more powerful on heavily constrained problem instances than any other known technique. Solutions to problems in excess of 45 customers (equivalent to a 91 city TSP) are solved on an IBM 3031 computer in a

matter of seconds. The efficiency is achieved by only generating feasible state space vectors, thus greatly reducing the storage and execution requirements. The same dynamic programming algorithm is also used to solve the multiple vehicle PUDP but with less impressive results. Other exact techniques could not be effectively used on the PUDP due primarily to the precedence requirement.

Heuristic algorithms were also developed and tested. Most of the algorithms commonly used to solve the related TSP and PUDP perform poorly on the PUDP, often producing solutions as much as 50% above optimal. An interchange (3-optimal) heuristic consistently produced superior results for the single vehicle PUDP. Solutions averaging within 1% of optimal were obtained for heavily constrained problem instances.

The multiple vehicle problem is significantly more complex than is the single vehicle problem. Results for the multiple vehicle problem were acceptable but inconclusive. Consequently, the multiple vehicle area is judged to be the most promising area for future research.

AFIT RESEARCH ASSESSMENT

81-50D

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (ATC). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

RESEARCH TITLE: The Single and Multiple Vehicle Pickup and Delivery Problem: Exact and Heuristic Algorithms

AUTHOR: Gerald R. Armstrong

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?

( ) a. YES ( ) b. NO

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

( ) a. YES ( ) b. NO

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

( ) a. MAN-YEARS ( ) b. $

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?

( ) a. HIGHLY SIGNIFICANT ( ) b. SIGNIFICANT ( ) c. SLIGHTLY SIGNIFICANT ( ) d. OF NO SIGNIFICANCE

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME GRADE POSITION

ORGANIZATION LOCATION

STATEMENT(s):

THE SINGLE AND MULTIPLE VEHICLE PICKUP AND DELIVERY PROBLEM: EXACT AND HEURISTIC ALGORITHMS

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Gerald R. Armstrong

June 1981

ABSTRACT

The pickup and delivery problem (PUDP) represents a class of sequencing or routing problems where the key facet of the routing is that a pickup must precede the corresponding, subsequent delivery. Other considerations such as service time windows, quality of service parameters or operational constraints on either the driver or the vehicle are possible. As such, the PUDP is a constrained version of the ubiquitous travelling salesman problem (TSP), which seeks a minimum cost route that from an initial point visits each city or stop once and only once, ending at the initial stop. There are also similarities between the PUDP and the much studied vehicle routing problem (VRP), although the two problems are distinctly different because of the origin preceding destination requirement.

The TSP and VRP literature is extensive, offering both theory and algorithms for the solution of these problems. Given the similarities of these problems to the PUDP, those algorithms that performed well on TSP's and VRP's are discussed in detail and served as the basis for developing both exact and heuristic algorithms to solve the PUDP.

Assuming that all problem constraints are expressable in terms of stop numbers along a vehicle's route, dynamic programming can be used to optimally solve the problem. The algorithm developed is significantly more powerful on heavily constrained problem instances than any other known technique. Solutions to problems in excess of 45 customers (equivalent to a 91 city TSP) are solved on an IBM 3031 computer in a

matter of seconds. The efficiency is achieved by only generating feasible state space vectors, thus greatly reducing the storage and execution requirements. The same dynamic programming algorithm is also used to solve the multiple vehicle PUDP but with less impressive results. Other exact techniques could not be effectively used on the PUDP due primarily to the precedence requirement.

Heuristic algorithms were also developed and tested. Most of the algorithms commonly used to solve the related TSP and PUDP perform poorly on the PUDP, often producing solutions as much as 50% above optimal. An interchange (3-optimal) heuristic consistently produced superior results for the single vehicle PUDP. Solutions averaging within 1% of optimal were obtained for heavily constrained problem instances.

The multiple vehicle problem is significantly more complex than is the single vehicle problem. Results for the multiple vehicle problem were acceptable but inconclusive. Consequently, the multiple vehicle area is judged to be the most promising area for future research.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER PAGE

CHAPTER PAGE

CHAPTER PAGE

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

OVERVIEW OF THE PICKUP AND DELIVERY PROBLEM

I. PROBLEM CLASSIFICATION

The pickup and delivery problem (PUDP) represents a class of sequencing or routing problems where the key facet of the routing is that a pickup must precede the corresponding, subsequent delivery. Many other constraints are possible based on the particular application. As such, the pickup and delivery problem is a constrained version of the basic, much studied travelling salesman problem (TSP). The travelling salesman problem seeks to find a minimum cost path that from an initial point, visits each city or stop once and only once, ending at the initial stop. More rigorous definitions are provided below and in Chapter II.

The vehicle routing problem (VRP) is also a constrained version of the TSP. In the vehicle routing problem the key consideration is vehicle capacity, although, as with the PUDP, other constraints may be applicable. It is not correct to classify the PUDP as a further constrained vehicle routing problem. Vehicle capacity need not be a factor in the PUDP. Given that capacity is a consideration, the manner in which it affects the problem is significantly different between the two problems. Consequently, it appears the proper classification is to treat both problems as constrained versions of the TSP.

The PUDP is representative of several practical routing situations. Examples include dial-a-ride services and courier services. Notwithstanding, few articles devoted to the PUDP have appeared in the published literature. This is in sharp contrast to the literature devoted to the TSP and the VRP which is voluminous and extends over the last quarter century. All evidence suggests that the PUDP is a relatively unexplored subject area.

II. PROBLEM DESCRIPTION

The dial-a-ride service (DARS) offers a convenient means of conceptualizing the PUDP. Suppose that an organization provides transportation services for the handicapped. Vehicles must pick up these people at their individual origins and take them to their destinations. Return trips sometime later in the day are also possibilities. The objective is to satisfy all requests for service as economically as possible. In addition to the origin-destination (O/D) constraint, other intuitively appealing constraints may include:

1. A limitation on the number of passengers who can occupy the vehicle at any one time (capacity constraint);

2. A limitation on the amount of time that any one passenger must remain in the vehicle (quality of service constraint);

3. A range of times in which pickup and/or delivery must be made (time window constraint);

4. A requirement that the same driver provide both legs of a person's round trip (stop-back constraint);

5. Limits on the total distance that a vehicle may be driven (operational constraint).

III. PROBLEM SIGNIFICANCE

As mentioned above, the PUDP is representative of real world routing situations. Fisher and Jaikumar (10) estimated that urban delivery vehicles in 1975 travelled approximately 70 billion miles at a fuel cost in excess of $5.5 billion. Fuel costs have more than doubled in the last five years. Consequently, annual fuel costs for urban delivery vehicles in excess of $10 billion is clearly probable. The percentage of urban vehicles engaged in services that could be classified as fitting the pickup and delivery model is not known. However, were the figure as low as 5%, a figure in excess of $500 million is obtained.

Dial-a-ride services are playing an increasingly important role in urban public transportation. Such services for the handicapped, or transportationally disadvantaged, are available in nearly every American city, either provided by the public, by charitable organizations, or by volunteers. Unquestionably, a 5% improvement in route efficiency would produce a tremendous savings in both dollars and barrels of oil.

Experience with the vehicle routing problem suggests savings of from 5% to 10% are possible by applying fairly simple computer assisted algorithms to route the vehicles. Because the PUDP is inherently more complex, it appears less likely that a dispatcher, acting without

benefit of some algorithm, could produce a good route. Consequently, even a greater savings percentage appears possible for the PUDP.

IV. RELATIONSHIP TO THE TSP AND THE VRP

There are similarities as well as differences among the PUDP, the TSP, and the VRP. Since the TSP and VRP have been extensively investigated, exploitation of the similarities was a logical course of action.

Let G = (N,A,C) be a complete network with N representing the set of nodes, A the set of arcs, and C = $[c_{ij}]$ a matrix of costs representing the cost of going from node i to node j. A Hamiltonian cycle is a cycle that passes through each node $i \in N$ exactly once. The TSP is the problem of finding a least cost Hamiltonian cycle on G. The multiple travelling salesman problem (MTSP) requires that for m salesmen one find m cycles on G such that every $i \in N$ is visited exactly once and the total cost of the m cycles is minimal.

If we further constrain the MTSP by requiring that for any cycle or tour the capacity (in weight, volume...) of the corresponding vehicle cannot be exceeded, the resulting problem is the VRP. Although other constraints, such as time-windows or stop-backs, are possible, very few articles address them even in passing. One notable exception is the paper by Fisher and Jaikumar (10), which explicitly considers the time-window constraints.

In none of these problem definitions is there any mention of a requirement that one stop be visited before another. Incorporating this procedence relationship, required to define the PUDP, complicates

the TSP or MTSP much more than does capacity. The PUDP is considered more complicated and at least as difficult as any of these related problems.

V. PROBLEM DIFFICULTY

The TSP has been shown to be NP-complete (12). One consequence of this classification is that there is no known polynomial time algorithm that optimally solves it, despite hundreds of man-years devoted to finding one. If one could find a polynomial time algorithm for any one of the more than 300 NP-complete problems, one could solve all problems in NP in polynomial time. As the following lemma shows, the PUDP is at least as hard as the TSP. If a polynomial time algorithm exists that solves the PUDP, it could be used to solve the TSP in polynomial time. This in turn implies that it could be used to solve all of the other NP-complete problems. Consequently, the likelihood of anyone finding a polynomial time algorithm for the PUDP is not considered high.

Lemma 1

Unless P = NP, there does not exist a polynomial time algorithm that optimally solves the general PUDP.

Proof

It will be shown that one instance of the PUDP can be reduced to two travelling salesmen problems. Since the TSP is NP-complete and

cannot be solved in polynomial time unless P = NP, this implies that the PUDP can not be solved in polynomial time unless P = NP.

Consider the instance where all origins are to be visited before lunch, all destinations after lunch, and the driver must return to the depot for lunch. Clearly the optimal solution for the PUDP is the optimal sequencing over the set of origins coupled to the optimal sequencing over the set of destinations. But these optimal solutions are the TSP solutions taken over their respective sets. Hence, unless P = NP, a polynomial time algorithm for the general PUDP does not exist.

VI. RESEARCH QUESTIONS

The TSP and VRP literature offers an abundance of theory and algorithms for the solution of their respective problems. Given the similarities between the PUDP and the VRP and TSP, one would suspect that many of the same algorithms could be used for the PUDP. To what extent and how well remained to be determined. Consequently, this research was guided by the following three general questions:

1. Using what algorithms, and under what conditions, can the PUDP be efficiently solved optimally?

2. How well do heuristics commonly applied to related problems perform?

3. Which heuristics (s) provide the best PUDP solutions?

VII. ORGANIZATION

Chapter II provides a review of the literature pertinent to the pickup and delivery, travelling salesman, and vehicle routing problems. Those algorithms, both exact and heuristic which have been successfully applied to the TSP and the VRP are developed in detail.

In Chapter III, a detailed mathematical formulation for the PUDP is presented. The formulation provides the insight necessary to explain why some of the algorithms developed in Chapter II do not work efficiently when applied to the PUDP, while others do.

Chapter IV deals with the optimal solution to the single vehicle problem. A dynamic programming solution is developed which is extremely powerful when applied to highly constrained PUDP instances. The same dynamic programming model is also used to solve the multiple vehicle PUDP discussed in Chapter VI. Exact solutions other than by the dynamic programming approach are shown to be much more difficult to obtain for the PUDP than for the TSP.

Single vehicle heuristics are discussed in Chapter V. Many of the heuristics that are widely applied to the VRP are shown to perform poorly on the PUDP, especially when side constraints become more binding. Special attention is given to those instances of the PUDP for which the dynamic programming technique provides an optimal solution. This allows for a precise evaluation of how well a given heuristic performs. Only relative performance has previously been possible for all but extremely small vehicle routing problems.

The multiple vehicle PUDP, discussed in Chapter VI, is seen to be a most difficult problem. The precedence requirements render inefficient the heuristic determined to be the most powerful for the single vehicle problem. Further, the additional alternatives available due to more than one vehicle being available for customer assignment destroy the efficiency of the dynamic programming technique. Heuristics that proved somewhat successful as well as those that failed are detailed. When failure is encountered, an explanation is offered.

During the period of the research, many areas ripe for research were encountered. Practical limitations on available time precluded investigation of these opportunities as a part of this effort. Chapter VII, therefore, details several of the more interesting areas remaining to be explored.

CHAPTER II

LITERATURE REVIEW

The pickup and delivery problem (PUDP) has been classified as a constrained version of the travelling salesman problem (TSP). The vehicle routing problem (VRP) is also a constrained TSP. Any attempt to solve the PUDP must, therefore, logically begin with an examination of those algorithms successfully used in solving these two related problems. This chapter summarizes the results of such an examination. The few articles directly relating to the PUDP are also discussed.

The travelling salesman problem and the related vehicle routing problem are two of the most studied problems in operations research. Literally hundreds of algorithms, many representing minor modifications of others, have been proposed for their solution. It would neither be practical nor useful to attempt to address all of these algorithms. Rather, a more useful approach suggests discussing the basic approaches and underlying concepts of those techniques that have shown the greatest success in solving the TSP.

I. MATHEMATICAL FORMULATION

Travelling Salesman Problem

A word description of the TSP was presented in Chapter I. Specifically, the TSP seeks to minimize

$$\sum_i \sum_j c_{ij} x_{ij} \tag{2.1}$$

subject to:

$$\sum_i x_{ij} = 1 \quad , \quad j = 1,2,\ldots,n \tag{2.2}$$

$$\sum_j x_{ij} = 1 \quad , \quad i = 1,2,\ldots,n \tag{2.3}$$

$$y_i - y_j + nx_{ij} \leq n - 1 \quad , \quad i \neq j = 2,3,\ldots,n \tag{2.4}$$

$$x_{ij} \in \{0,1\} \tag{2.5}$$

$$y_i \text{ arbitrary real number} \quad , \quad i = 1,2,\ldots,n \tag{2.6}$$

where,

c_{ij} = cost of going directly from city i to j, and

$$x_{ij} = \begin{cases} 1, & \text{if the salesman goes directly from city } i \text{ to city } j \\ 0, & \text{otherwise.} \end{cases}$$

Expressions (2.2) and (2.3) insure that the salesman visits and departs from each city exactly once. Expressions (2.4) and (2.6) are the subtour-elimination conditions derived originally by Miller, Tucker and Zemlin (26) and often quoted by others.

Another common expression for eliminating subtours is

$$\sum_{i \in Q} \sum_{j \notin Q} x_{ij} \geq 1 \tag{2.6.1}$$

for every nonempty proper subset Q of N, where $\bar{Q}$ is the complement of Q. Bellmore and Malone (2) showed that this formulation, which is of order 2^n leads to an effective solution algorithm by branch and bound. The branch and bound algorithm, as well as other solution approaches, is discussed later in this chapter.

Although the former formulation is more compact than the latter, both express the problem precisely in mathematical notation. The compact formulation is perhaps more elegant and may be of value computationally if the constraints must be explicitly considered. The less compact formulation is sometimes more advantageous if the constraints are implicitly handled. For the purpose of defining the TSP, either is satisfactory. For the research discussed later, there is no computational preference for one notation over the other. However, the more compact formulation is used throughout simply because it is notationally more elegant.

Multiple Travelling Salesman Problem

The multiple travelling salesman problem (MTSP) can be easily transformed into a standard TSP. Suppose there are M salesmen. M copies of the origin are included in the cost matrix $C = [c_{ij}]$, with each of the copies representing a unique stop but with the same costs relative to the other nodes as the origin has. The M copies are connected with arcs of infinite (or extremely large) cost so that it is never profitable to include one of these arcs in a solution. The resulting solution to the TSP taken over this expanded network provides

the solution to the MTSP. Consequently, subsequent discussion on the TSP is equally applicable to the more general MTSP.

Vehicle Routing Problem

The VRP formulation is similar to the TSP formulation except that it is necessary to include the number of vehicles, k, into the formulation primarily to define the capacity constraints, Q_k, for each vehicle. Each customer has a requirement, r_i. An integer programming formulation for the VRP is:

$$\text{minimize} \quad \sum_i \sum_j \sum_k c_{ij} x_{ij}^k \tag{2.7}$$

subject to

$$\sum_i \sum_k x_{ij}^k = 1 \quad , \quad j = 1,2,\ldots,n \tag{2.8}$$

$$\sum_i x_{ip}^k - \sum_j x_{pj}^k = 0 \quad , \quad \begin{matrix} k = 1,2,\ldots,K \\ p = 1,2,\ldots,n \end{matrix} \tag{2.9}$$

$$\sum_i \sum_j r_i x_{ij}^k \leq Q_k \quad , \quad k = 1,2,\ldots,K \tag{2.10}$$

$$\sum_i \sum_j c_{ij} x_{ij}^k \leq D_k \quad , \quad k = 1,2,\ldots,K \tag{2.11}$$

$$\sum_j x_{ij}^k = 1 \quad , \quad k = 1,2,\ldots,K \tag{2.12}$$

$$y_i - y_j + n \sum_k x_{ij}^k \leq n-1 \quad , \quad i \neq j = 1,2,\ldots,n \tag{2.13}$$

$$x_{ij}^{k} \in \{0,1\} \quad , \quad \text{for every } i, j \text{ and } k \tag{2.14}$$

$$y_i \quad \text{arbitrary real number,} \quad i = 1,2,\ldots,n \tag{2.15}$$

where

$$x_{ij}^{k} = \begin{cases} 1, \text{ if vehicle k visits customer j immediately after visiting customer i} \\ 0, \text{ otherwise.} \end{cases}$$

Many of these expressions are logical extensions of those for the TSP. Expression (2.9) requires that if a vehicle visits a customer, it must depart from this same location. Expressions (2.10) and (2.11) represent the capacity and operational limitations of the kth vehicle, while (2.12) insures that a vehicle must be used once and only once. The formulation given above is refered to hereafter as the standard VRP.

Other constraints, such as delivery windows, are seldom addressed at all. One notable exception is the paper by Fisher and Jaikumar (10), which explicitly considers the delivery time window constraints. Their formulation is much more complex and is, therefore, not included here.

II. EXACT ALGORITHMS

Finding exact solutions to all but relatively small (less than 50 cities) travelling salesman problems has proven to be a difficult task. Exact solutions to the vehicle routing problem are much more difficult to come by. Christofides was credited with claiming that the largest vehicle routing problem of any complexity that had been solved exactly

involved only 23 customers (18). Christofides subsequently reports optimal solutions of VRP's of up to about 30 customers (4). The reason for these difficulties is the exponential growth in computations required to guarantee optimality.

Four approaches for finding exact TSP solutions will be discussed. These techniques are integer programming by means of cutting planes, branch and bound based on subtour elimination, dynamic programming, and Lagrangean relaxation using minimum 1-trees. None of these approaches are new, the most recent introduced to the literature in 1970 (22). In addition, a Benders decomposition approach for solving the vehicle routing problem will be outlined.

Integer Programming

Other than the integrality conditions on x_{ij} in expressions (2.5) and (2.14) the above formulations allow for solution by ordinary linear programming (LP). Suppose these conditions are relaxed to

$$0 \leq x_{ij} \leq 1 \quad \text{for every } i,j \quad . \tag{2.16}$$

The LP solution will not generally be integral. However, it is possible to add additional constraints to the final LP tableau to eventually obtain integrality. These additional constraints are called cutting planes. Garfinkel and Nemhauser (13) provide a treatment of the theory of cutting planes. The original theory of cutting planes is due mainly to Gomory (19).

The basic idea of a cutting plane algorithm is to "cut away," using hyperplanes, the noninteger portions of the feasible convex hull of the relaxed linear program. These hyperplanes are constraints that can be generated at each step from the current LP tableau, and taken in such a manner that no feasible, integer solutions are ignored.

Finite cutting plane algorithms have been proposed and used to solve the TSP, as well as other integer programming problems. For the most part they have not performed well, the exception being the recent work of Miliotis (25). Fisher and Jaikomar (10) in their VRP algorithm use the Miliotis cutting plane algorithm to solve TSP subproblems with impressive results.

Branch and Bound

Branch and bound techniques are by far the most common type applied to the TSP, especially those employing a subtour elimination scheme. Expressions (2.1), (2.2), (2.3) and (2.5) taken separately define the assignment problem (AP) – the objective being the most efficient assignment of n men to n distinct jobs. The assignment problem, a relaxation of the TSP, is easily solved in polynomial time. Since the AP is a relaxation of the TSP, the optimal solution to the AP, which is most generally not feasible to the TSP, provides a lower bound on the optimal value of the TSP solution. Any feasible TSP solution provides an upper bound. If the AP solution is not feasible to the TSP, because of subtours, one branches into k subproblems, where k is the number of arcs in one of the subtours. In each subproblem, one of the k original

subtour arcs is assigned an infinite cost, this breaking or eliminating the subtour. The k new AP's are then solved, bounds computed, and the process repeated if subtours are present and the lower bound is less than the best feasible tour as yet found. Eventually, one is assured of finding the optimal solution, although the size of the branching tree may become enormous for large problems.

The approach outlined above represents one method of using branch and bound techniques to solve the TSP. The key to the success of a branch and bound algorithm rests with obtaining very tight bounds, thus greatly reducing the size of the branching tree, and with branching rules, which also minimize the resulting tree size. Branch and bound algorithms often find optimal or near optimal solutions toward the beginning of the enumeration. Thus, much of the time required by the algorithm is spent in verifying the optimality of a tentative solution. Consequently, branch and bound techniques can often be terminated early, producing a very good solution. In this sense, the technique is used as a heuristic. Most cutting plane algorithms do not have this characteristic.

Dynamic Programming

Dynamic programming has been used less on the TSP than some other techniques. It does not appear that dynamic programming has been used with any success on the VRP. The computer storage requirements are the primary limiting factor. Even though the dynamic programming recursions allow for treating combinations rather than permutations of the

operations required, the combinations increase exponentially with the size of the problem. Consequently, core storage requirements to solve a 20 city TSP exceed 900,000 words.

The basic dynamic programming recursion to determine a shortest partial TSP tour from the origin (node 1) to node j that passes through $i_1, i_2, \ldots, i_{k-1}$ is

$$f_k(j|i_1,i_2,\ldots,i_{k-1}) = \min_{m=1,\ldots,k-1} [f_{k-1}(i_m|i_1,i_2,\ldots,i_{m-1},i_{m+1},\ldots,i_{k-1}) + c_{i_m,j}] \quad (2.17)$$

By letting

$$S_k = \{i_1,i_2,\ldots,i_{k-1}\} \quad (2.18)$$

we can simplify (2.17) to obtain

$$f_k(j|S_k) = \min_m [f_{k-1}(i_m|S_k - i_m) + c_{i_m,j}) \quad . \quad (2.19)$$

The initial recursive equation thus becomes

$$f_2(j|i_1) = c_{1,i_1} + c_{i_1,j} \quad \text{for all } i_1 \text{ , } j \neq 1 \text{ and } i_1 \neq j, \quad (2.20)$$

while the final, which terminates with an optimal tour value, is obtained by solving

$$f_n(1|S_n) = \min_m [f_{n-1}(i_m|S_n - i_m) + c_{i_m,1}] \quad . \quad (2.21)$$

The number of f_k values is given by

$$g(n,k) = (n-1)! / (k-1)!(n-k-1)! \qquad (2.22)$$

which reaches a maximum halfway through the computations.

The procedure for determining the optimal tour is a two phase one. First, f_k, $k = 2,3,\ldots,n$ are computed recursively by (2.10). Then the optimum ordering $(i_1,i_2,\ldots,i_n)$ is obtained by picking the i_m such that (2.10) holds in decreasing order of k, $k = n, n-1, n-2,\ldots,2$.

Lagrangean Technique

One of the most powerful techniques for solving the symmetric TSP is the 1-tree approach of Held and Karp (22,23). A TSP is symmetric if

$$c_{ij} = c_{ji} \quad \text{for every } i,j \quad . \qquad (2.23)$$

A 1-tree is a tree taken over vertices $2,3,\ldots,n$, connected to vertex 1 with two edges. A TSP tour is a 1-tree for which each vertex has degree 2. Figure 1 depicts a 1-tree over 8 vertices. A minimum weight 1-tree can be found by first finding a minimum spanning tree over the vertex set $\{2,3,\ldots,n\}$, and then adding the two least cost edges at vertex 1. The minimum spanning tree is easily solved in order n time.

Changing the cost matrix for intercity distances by

$$\hat{c}_{ij} = c_{ij} + \pi_i + \pi_j \qquad (2.24)$$

does not alter the optimal solution to the TSP but does alter the

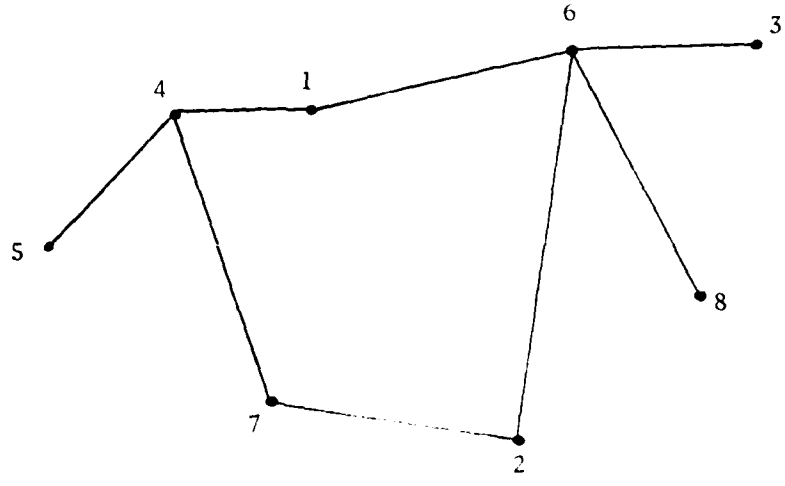


Figure 1. An example of a 1-tree.

solution of the minimum 1-tree. An iterative method for approaching the optimal solution from below is based on altering the π_i and π_j's so that each vertex is forced toward degree 2. Normally, the algorithm must employ branch and bound to obtain the final TSP tour. However, as Held and Karp note the bounds computed by the final minimum 1-trees ". . .are so sharp that the search trees are miniscule compared to those normally encountered. . . ." Consequently, the 1-tree approach could be considered a branch and bound algorithm that uses a Lagrangean relaxation to compute tight lower bounds. It is important to note that the technique is only valid for the symmetric travelling salesman problem.

Christofides, Mingozzi, and Toth have extended Held and Karp's Lagrangean relaxation concept to the VRP (4). It is this work for which solutions up to 30 customers are reported. The key factor allowing for success is again tight bounds computed by similar Lagrangean penalty or relaxation procedures.

Benders Decomposition

One solution technique (10) for the VRP, which specifically addresses delivery windows, employs Benders decomposition. The algorithm iterates between a generalized assignment problem (GAP) to assign customers to vehicles such that vehicle capacity is not exceeded, and a TSP subproblem for sequencing the stops. As mentioned above, a cutting plane algorithm is used to solve the TSP. Infeasible subproblems produce the Benders cuts needed to obtain delivery window

feasibility on subsequent iterations. Computational results remain incomplete, but appear promising.

Fisher and Jaikumar comment that although their algorithm will produce an optimal solution in a finite number of iterations, they expect that for practical size problems the algorithm will be used as a heuristic. This is accomplished by terminating the iterations prior to achieving optimality and using the best feasible solution found thus far.

The upper limit on the size of a general TSP that can be optimally solved, within practical limits on time and computer storage, by any known technique is questionable, but certainly is not greater than 100 cities and is probably less. For the VRP, this limit is significantly lower. For problem instances of such size that obtaining an optimal solution is impractical, heuristic solutions provide the only alternative.

III. HEURISTIC ALGORITHMS

A multitude of heuristic algorithms have been proposed for both the TSP and the VRP. Many heuristics for the VRP are just slight modifications of ones used for the TSP. The heuristics to be discussed fall into one of three broad classes: tour construction approaches, tour improvement approaches, and composite approaches. The latter is the logical combination of the first two. Each class, and the common representatives of each, will be examined.

Tour Construction Heuristics

Tour construction algorithms generate a tour, one stop at a time, from the given distance or cost matrix. When constraints are present, as in the VRP, the next stop is added to the partially constructed tour only if it can lead to a feasible final solution. With the VRP, this would normally be a check to insure that vehicle capacity had not been exceeded, a relatively simple procedure. Insuring feasibility with delivery windows is a much more complicated procedure and may, to some extent, explain why such constraints are not treated in the literature.

By far the most common of the tour construction procedures is the Clarke-Wright savings algorithm which dominates VRP solution attempts. Others include the nearest neighbor or greedy approach and various insertion approaches. Since the literature for the VRP indicate the general superiority of the Clarke-Wright models over other tour construction heuristics, they will be discussed first.

Clarke-Wright savings. The basic concept of the travel time saved heuristic was developed by Clarke and Wright (7), who credited the earlier work of Dantzig and Ramser (8). In both cases, the procedure was developed to solve the VRP. Numerous modifications have been suggested, but the underlying concept remains invariant (11, 28, 38, 42, 43).

The procedure begins by selecting one node as the origin. With the VRP, no choice is required. Initially one assumes that every stop is visited directly from the origin. Then the savings that can be

achieved by combining two subtours into one, by linking stops i and j, is computed by

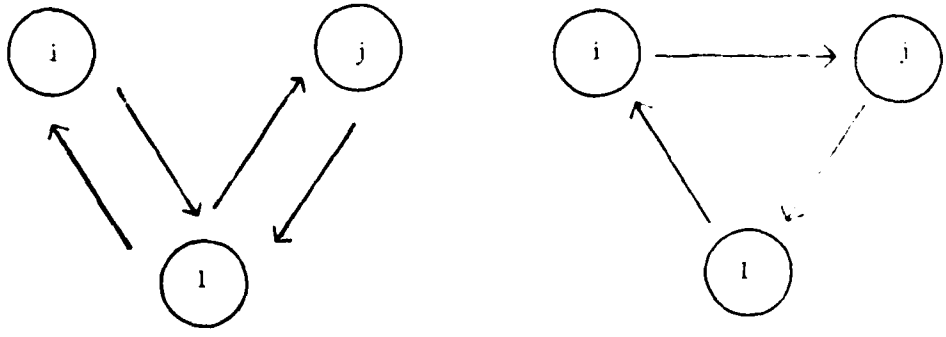
$$s_{ij} = c_{1i} + c_{1j} - c_{ij} \quad . \tag{2.25}$$

Beginning with the largest of these savings values, routes are assembled such that the next stop added has the largest remaining savings — provided that a constraint is not violated. Customers that have been linked are treated as a single macro customer. Figure 2 demonstrates the algorithm for two stops i and j. Once a pair of stops has been linked, they remain linked. The algorithm continues until all stops have been assigned.

Most commercially available algorithms for routing vehicles are based on the Clarke-Wright savings concept. Survey results indicate a percentage in excess of 80%, including the IBM code, VSPX, which is the most used package in America (18). Empirical evidence suggests that, on average, the Clarke-Wright savings method produces vehicle routes that are as good as, and often better than, routes produced by other heuristics for the standard VRP. Christofides and Eilon (6) obtained an average deviation of 3.2% from optimal on 10 small vehicle routing problems.

One of the attempts to improve the Clarke-Wright model is by modification of the savings equation. Yellow (43) suggested a route shape parameter θ such that the equation is

$$s_{ij} = c_{1i} + c_{1j} - \theta c_{ij} \quad . \tag{2.26}$$



(a) Before linking (b) After linking

Figure 2. Concept of the Clarke-Wright savings heuristic.

Varying the parameter ν alters the emphasis placed on the cost between stops i and j in relation to their costs relative to the central depot. When $\nu = 1$, Yellow's model reduces to the Clarke-Wright model. Typically, several values of ν are tried, including $\nu = 1$. Therefore, Yellow's method always produces at least as good result as the Clarke-Wright technique.

Tillman and Cochran (39) and Tillman and Cain (38) suggested another approach based on extending the savings calculations to more than one stop in the future. Although their combined reported works are limited to two choices in sequence, the concept is extendable to three or more in sequence. However, the possible combinations are of order n^k, where k is the number of positions examined. Therefore, this procedure rapidly becomes expensive in terms of computational effort.

Nearest neighbor. The nearest neighbor procedure is the simplest and perhaps most naive of all heuristics suggested. One always selects the nearest unvisited, feasible stop until the tour is completed. Because of the way in which the algorithm works, it is often referred to as a greedy algorithm. The primary appeal of the greedy approach is its simplicity. It is easily understood and large problems can easily be solved by hand. Unfortunately, the greedy approach does not normally produce "good" tours for either the TSP or the VRP.

Insertion procedures. An insertion procedure uses a specific selection rule to pick a stop not yet in the partial solution and then determines where to insert this stop. Three common types of selection

criteria for the next stop to be included are nearest, farthest, and random, with reference to any one of the nodes included in the partial tour. Insertion is accomplished for node k by finding arc (i,j) in the subtour which minimizes

$$c_{ik} + c_{kj} - c_{ij} \tag{2.27}$$

subject to the problem constraints, if any. Computational experience indicates results within 3-5% of optimal or best known solution are attainable with insertion algorithms (17).

Tour Improvement Heuristics

The basic idea behind tour improvement procedures is to take a given feasible tour and systematically modify the tour to obtain a better one. The procedure is one of arc interchange. Two related heuristics have worked well on the TSP and the VRP: the r-optimal heuristic and the λ-optimal heuristic of Lin and Kernighan (24).

r-optimal. The concept of r-optimality is an outgrowth of research on the TSP. Christofides and Lilon (6) appear to have been the first to apply the concept to the vehicle routing problem, with results at least as good as those obtained using the Clarke-Wright approach. The term r-optimal implies that no improvement in a given feasible solution is possible by eliminating any r links and replacing them by r new links. In other words, the procedure terminates at a local optimum. The r-optimal procedure is an $O(n^r)$ algorithm. Consequently,

only 2-optimal and 3-optimal interchanges have been used for the TSP and VRP. Larger values of r could be used and would result in at least as good a solution since r optimality implies r-1 optimality. However, the increased computational cost could apparently not be justified.

Christofides and Eilon suggest that a 3-optimal algorithm first find a 2-optimal tour and then use this tour as the input to the 3-optimal algorithm. Figure 3 shows the only legitimate reconnection pattern for the 2-optimal algorithm and the five reconnection patterns for the 3-optimal algorithm which exclude duplication of a 2-optimal pattern. In Figure 3, the letters represent the arcs that are removed and the numbers represent the specific stops that are to be reconnected. Initially, the stops are visited in numerical sequence, with intervening stops unnumbered. The heavy lines represent all intervening stops between each of the two endpoints; the dotted lines the single arc removed; and the remaining lines the new arcs. The patterns do not depend on whether the network is directed or undirected. However, in the directed case, some of the patterns may be infeasible due to the nonexistence of an arc in the reverse direction.

λ-optimal. Closely related to the r-optimal procedure is the λ-optimal technique of Lin and Kernighan. The λ-optimal heuristic was developed to solve the TSP and does so with remarkable results. The probability of optimally solving a given 40 city TSP is reportedly close to one, while an optimal solution to a 100 city TSP is reported at close to 25%.

(a) 2-optimal

(b) 3-optimal

Figure 3. Two and 3-optimal reconnection patterns.

The λ-optimal approach works essentially in the same manner as does the r-optimal approach except that λ is not fixed at any step of the iteration process. Rather, for each iteration, λ starts at a value of 2, i.e., two arcs are initially removed and a 2-optimal solution found. Then a third arc is removed and a 3-optimal solution found. Additional arcs are incrementally removed until no further feasible improvement in total tour cost can be made. Consequently, four, five or even more arcs may be removed at one time. With five arcs removed, the algorithm is searching for a better solution than the current 4-optimal one. The algorithm insures that if an arc is removed, a feasible reconnection pattern does exist, thus precluding investigation of profitable, but infeasible reconnection patterns. The creation of separate subtours is an example of an infeasible reconnection pattern.

Composite Heuristics

A composite procedure constructs an initial tour using one of the tour construction techniques and then attempts to improve on this solution using a tour improvement technique. Since the composite will always do at least as well as the tour construction algorithm without improvement, the question of whether to use a composite is one of accuracy desired and resources available. Several unique composite approaches have been suggested, specifically for application to the VRP. Two of these are of special interest to the material developed in later chapters.

Sweep algorithm. The sweep algorithm, developed by Gillett and Miller (15), employs the Lin-Kernighan algorithm to sequence stops on cluster generated routes. Other models of the vehicle routing problem can be characterized as solving one big problem. Gillett and Miller use a technique which divides the big problem into k subproblems, where k is the number of vehicles.

The algorithm orders all delivery points sequentially by their polar coordinate angle and then selects routes by sweeping through the angles. Points are not added if one of the constraints would be violated. An exchange routine is applied to the resultant route structure to obtain any additional route improvement. The sweep and improvements are accomplished in both a forward and a backward direction with the best result taken as the final solution. Gillett and Miller recommended and used the Lin-Kernighan algorithm for the improvement step. They claim that problems well in excess of 100 customers are computationally feasible with their sweep algorithm, and solutions are very competative with other solution approaches.

MTOUR. MTOUR is the name given to an algorithm developed by Russell (36) to solve constrained multiple travelling salesman problems or constrained VRP's. The additional constraint is a delivery window or a due date requirement. Russell solves the multiple problem as a single problem on an extended network, as discussed above, using the Lin-Kernighan approach.

As with all of the tour improvement heuristics, an initial feasible solution is required. In the more heavily constrained environment, such a starting solution may not be obvious. In the more constrained problem, the feasibility of arc interchanges greatly complicates the λ-optimality procedures. Notwithstanding, Russell is one of the few who does address the time window constraint and he reports good success in using his MTOUR on his test problems.

The heuristics discussed here are not intended to be exhausive. However, they do represent the basic concepts and approaches that have been successfully applied to the TSP and to the VRP. Articles devoted specifically to the class of problems described by the pickup and delivery problem are far less numerous.

IV. PUDP RELATED LITERATURE

Two articles and one working paper discuss subject areas that fall within the framework of the PUDP model. Only the article by Psaraftis (34) is considered significant to the research effort. A brief summary of the other two is presented before Psaraftis's article is discussed in more detail.

Single Vehicle PUDP

In their 1979 unpublished paper, Driscoll and Emmons (9) discuss routing of a single vehicle to meet pickup and delivery requirements. The only problem constraint is that a pickup must precede a subsequent delivery. The vehicle is taken to have unlimited capacity. Driscoll

and Emmons propose a heuristic that employs a 2-optimal algorithm. All origins are 2-optimal sequenced, and then all destinations are 2-optimal sequenced. Finally, the two lists are joined, and a final 2-optimal sequence is formed.

Dial-a-Ride Systems

Stein (37) presents an investigation of the quality of any proposed heuristic in a theoretical, asymptotic, probabilistic sense. No specific solution is offered. All of the results cited implicitly assume vehicles of unlimited capacity.

Dynamic Programming and Dial-a-Ride

The May 1980 edition of Transportation Science includes the article by Psaraftis that outlines the use of dynamic programming to optimally solve the single vehicle, many-to-many, immediate-request dial-a-ride problem. This article appeared subsequent to the research reported in latter chapters, and in no way influenced that work.

The term many-to-many is used to imply that the pickup points and the delivery points of the individual customers are all distinct points. Immediate-request is used to imply that every customer wishes to be serviced as soon as possible. The article addresses both the static and the dynamic cases. Static implies that no additional requests for service can be satisfied by the single vehicle once it begins servicing customer requests.

Objective. Psaraftis uses an objective function that is a weighted combination of total service time for all customers and of the total service time for all customers and of the total dissatisfaction experienced by customers in waiting to be picked up and awaiting delivery at their respective destinations. A more conventional objective function could be substituted without altering the algorithm. A maximum position shift (MPS) limits the absolute difference between the order in which a customer is served on the route developed and the position a customer held on a first come first serve list. This "position" orientation to the problem is well suited for solution by dynamic programming. The precedence, MPS and capacity constraints are all taken care of as the recursive procedure develops.

State vector. To define the state space, a state vector $(L,k_1,k_2,\ldots,k_N)$ is used where L represents the point or stop the vehicle is currently visiting, N is the number of customers and k_j is the status of the j^{th} customers such that

$k_j = 3$: customer j has yet to be picked up,

$k_j = 2$: customer j is in the vehicle, and

$k_j = 1$: customer j has been delivered.

A series of screening steps are used to determine if a particular state vector is feasible or not. One of the screening procedures requires, for each state, examining subsequent or next states reachable from the current state vector in order to determine if the present vector can be feasibly extended. The current state vector, tentatively classified as

feasible, is reclassified as infeasible if all subsequent states are determined to be infeasible. Consequently, much of the computational requirement involves these screening procedures.

Storage requirements. Psaraftis' dynamic programming algorithm requires a minimum of $2\ N\ 3^{N-1}$ storage locations, where N is the number of customers. For example, for 15 customers a total of over 430 million storage locations would be required. As actually implemented, Psaraftis' algorithm would require in excess of 1 billion storage locations to solve a 15 customer problem.

Computational results. The algorithm runs as an exponential function of the size of the problem (number of customers), but is shown to be asymptotically more efficient than dynamic programming applied to the travelling salesman problem. The largest problem reported on involved nine customers or 18 specific stops and required nearly 600 seconds to solve. Slight improvements in running times were noted as the MPS and capacity values were decreased, thus making the problem more heavily constrained. The amount of improvement between an unconstrained problem and a fully constrained problem (pickup followed by immediate delivery of each customer in first come first serve order) was less than 75%.

The algorithm as constructed can only be used to solve a single vehicle dial-a-ride problem. It appears to be practically limited to problems of nine customers or less due to storage and execution requirements. In Chapter IV, a different dynamic programming algorithm is

developed. This dynamic programming algorithm is much more powerful than Psaraftis' is on the more heavily constrained problem. Solutions to problems of well over 50 customers are readily attainable. The algorithm also can be used to solve the multiple vehicle problem.

CHAPTER III

MATHEMATICAL FORMULATION

I. MEMORY REQUIREMENT

In Chapter I a word picture of the pickup and delivery problem (PUDP) was presented. The key problem facet was seen to be the precedence relationship, requiring an origin be visited before its corresponding destination. Chapter II includes the mathematical formulations of the related travelling salesman and vehicle routing problems. These formulations are not directly extendable to describe the PUDP. The precedence relationship and any capacity restrictions preclude such an extension.

In order to define the precedence relationship as well as many of the other suggested, possible constraints for the PUDP, it is necessary to include in the model a memory to keep track of the sequence or timing of the stops. Identifying the sequence of stops is required to insure that a destination is not sequenced before its origin. Guaranteeing that time windows, quality of service standards and especially the vehicle capacity limitations are met, are examples of the other constraints that are time or sequence dependent. The memory provision can be achieved by incorporating integral time periods into the model. The time period approach is mandatory if an integer linear programming formulation to include the capacity constraint is desired.

II. NOTATION

The following notation will be employed through the remainder of this chapter:

L - number of nodes. L equals the number of origins plus number of destinations *plus one for each* vehicle's starting and ending point. Each item's or person's origin and destination is assigned a unique node. Consequently, if one specific location were to serve as a destination and a multiple origin of multiplicity m, with each origin having a different destination, m+1 distinct nodes would be required. There would be zero cost between each of these m+1 nodes. The depot from which all vehicles are dispatched is assigned stop number 1.

n - number of origin/destination pairs.

n = L/2 - # vehicles.

N = n+1 - the total number of distinct points.

O - set of origins.

D - set of destinations.

K - number of vehicles.

M_k - capacity of the kth vehicle.

R = [r] - a vector whose ith component represents the amount or quantity to be moved from origin i to destination 1+n. We assume $r_i = 1$ $i \in O$ for the DARS.

c_{ij} - cost of direct travel from stop i to stop j. The cost can be distance, time or other suitable measure.

τ_{ij} - travel time from stop i to stop j. Since travel time may be a valid measure of cost, it is possible that $c_{ij} = \tau_{ij}$ for all i and j. However, it is not necessary that any specific relationship exist between these values.

e_i - earliest time that stop i (either a customer's origin or destination) can be serviced. The value can either be expressed in terms of clock time or by a period number.

ℓ_i - latest time that stop i can be serviced, expressable in same manner as e_i.

Q_j - maximum allowable time between pickup and delivery for the jth customer requirement. Q_j can either be expressed in clock time or by the number of intervening periods.

D_k - upper limit on the distance or other appropriate measure of operational limits for the kth vehicle.

x^k_{ijt} - $\begin{cases} 1, \text{ if vehicle k proceeds directly from stop i to stop j,} \\ \text{where j is the vehicle's tth stop.} \\ 0, \text{ otherwise.} \end{cases}$

τ_i - clock time at which customer stop i is visited. It is assumed that the starting time is assigned a zero time.

S - a scalar taken to be larger than the length of any feasible tour.

T - set of time periods.

III. TOUR CONTINUITY CONSTRAINTS

As with the multiple travelling salesman and the vehicle routing problems, the total of all individual tours must incorporate all N stops. Because of the parameter t in the formulation, these constraints can be expressed more compactly than they can be by using the more conventional constraints. As before, compactness is preferred only for mathematical elegance.

Conventional Constraints

Expressions similar to those for the travelling salesman and vehicle routing problems can be used to force tour continuity for the PUDP. The problem is

$$\text{minimize} \quad \sum_{ijtk} c_{ij} x^k_{ijt} \tag{3.1}$$

subject to

$$\sum_{itk} x^k_{ijt} = 1 \quad , \quad j = 1,2,\ldots,N \tag{3.2}$$

$$\sum_{j,t=1}^{N} x^k_{rjt} - \sum_{i,t=1}^{N} x^k_{irt} = 0 \quad , \quad \begin{matrix} r = 1,2,\ldots,N \\ k = 1,2,\ldots,k \end{matrix} \tag{3.3}$$

$$\sum_{j=2}^{N} x^k_{1jl} \leq 1 \quad , \quad k = 1,2,\ldots,k \tag{3.4}$$

$$y_i - y_j + N \sum_{k=1}^{N} \sum_{t=1}^{N} x^k_{ijt} \leq N - 1, \; i \neq j = 1,2,\ldots,N \tag{3.5}$$

$$x^k_{ijt} \in \{0,1\} \qquad \text{for all } i,j,t \text{ and } k \tag{3.6}$$

$$y_i \quad \text{arbitrary real number.} \tag{3.7}$$

Expression (3.2) requires that every stop be visited exactly once, while (3.3) states that if vehicle k visits a given stop, it must also depart from it. Expression (3.4) insures that no vehicle is used more than once. The subtour elimination equations (3.5) are an extension of those proposed by Miller, Tucker and Zemlin for the the travelling salesman problem (26). The number of decision variables is of order KN^3. It should be noted that with the exception of the addition of the period subscript on the decision variable x, (3.1) to (3.7) are identical to those defining the vehicle routing problem. In fact, these tour continuity expressions could be expressed without including periodicity. Periodicity is required in defining other constraints, however, as will be seen later.

Compact Continuity Constraints

A paper by Fox, Gavish and Graves (11) provides a compact formulation of order N for the time dependent travelling salesman problem (TDTSP). The TDTSP is a variation of the TSP in which $C = [c_{ijt}]$. The cost of going from i to j depends on the time period t. It is assumed that travel time between any two cities is one time period. Clearly if C is invariant over t, the TDTSP reduces to the TSP. The TDTSP formulation of (11) can be expanded to define the tour

continuity constraints for the PUDP. The problem is

$$\text{minimize} \quad \sum_{ijtk} c_{ijt}\, x^k_{ijt} \tag{3.8}$$

subject to

$$\sum_{ijtk} x^k_{ijt} = L \tag{3.9}$$

$$\sum_{j=1}^{N}\sum_{t=2}^{N}\sum_{k=1}^{K} t\, x^k_{ijt} - \sum_{j=1}^{N}\sum_{t=1}^{N}\sum_{k=1}^{K} t\, x^k_{jit} = 1 \quad , \quad i = 2,3,\ldots,N \tag{3.10}$$

$$\sum_{j,t=1}^{N} x^k_{rjt} - \sum_{i,t=1}^{N} x^k_{irt} = 0 \quad , \quad \begin{matrix} r = 1,2,\ldots,N \\ k = 1,2,\ldots,K \end{matrix} \tag{3.11}$$

$$\sum_{j=2}^{N} x_{1j1} \leq 1 \quad , \quad k = 1,2,\ldots,K \tag{3.12}$$

$$x^k_{ijt} \in \{0,1\} \quad \text{for all } i,\ j,\ t \text{ and } k \quad . \tag{3.13}$$

These equations are of order NK, while the conventional expressions require equations of order N^2. As before, the number of decision variables is of order KN^3. Expressions (3.11) and (3.12) are analogous to (3.3) and (3.4). The contribution of Fox, Gavish and Graves is that expressions (3.9) and (3.10) effect subtour elimination.

IV. ORIGIN/DESTINATION CONSTRAINTS

The key element that distinguishes the pickup and delivery problem from the TSP and the VRP is the precedence requirement of origin before destination. This requirement can be expressed as

$$\sum_{i=1}^{N} \sum_{t=1}^{N} t\, x^{k}_{i,j+n,t} - \sum_{i=1}^{N} \sum_{t=1}^{N} t\, x^{k}_{ijt} \geq 1 \quad ,$$

$$\text{for every } j \in 0 \quad . \tag{3.14}$$

Were one willing to include the decision variable τ in the formulation, then

$$\tau_j < \tau_{j+n} \qquad \text{for every } j \in 0 \tag{3.15}$$

would express the same requirement. The latter expression does not require the periodicity parameter t, but does increase the number of decision variables.

Unless K = 1, it is also necessary to insure that the vehicle visiting the origin is the same vehicle that visits the corresponding destination. This can be accomplished by

$$\sum_{i=1}^{N} \sum_{t=1}^{N} x^{k}_{i,j+n,t} - \sum_{i=1}^{N} \sum_{t=1}^{N} x^{k}_{ijt} = 0 \qquad \begin{array}{l}\text{for every } j \in 0 \\ k = 1,2,\ldots,K\end{array} \quad .$$

(3.16)

V. VEHICLE CAPACITY CONSTRAINTS

The impact of vehicle capacity on the PUDP is totally different than it is on the VRP. For the VRP, a given set of customers either can or can't be serviced by a given vehicle, depending on the sum of the customer's requirements. With the PUDP, it is the sequence of stops which determines the quantity on the vehicle at any time. This fact is what necessitates the use of the period parameter t. The capacity of a vehicle is not exceeded at any time so long as

$$\sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=n+2}^{N} r_{j-n} x_{ijt}^{k} - \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=n+2}^{N} r_{j} x_{ijt}^{k} \leq M_{k}$$

$$\text{for every } T \in \bar{T}$$
$$k = 1,2,\ldots,K \qquad (3.17)$$

VI. TIME WINDOW CONSTRAINTS

There are two approaches to defining time window constraints. One approach employs clock time, while the other uses the periodicity parameter. For the former, the expressions are

$$\tau_i \geq e_i \quad , \quad i = 2,3,\ldots,N \qquad (3.18)$$

and

$$\tau_i \leq \ell_i \quad , \quad i = 2,3,\ldots,N \quad . \qquad (3.19)$$

For many instances of the problem, it is possible to approximate clock time by a stop number. For these cases

$$\sum_{t=e_j}^{\ell_j} \sum_{i=1}^{N} \sum_{k=1}^{K} x_{ijt}^{k} = 1 \quad , \quad j = 2,3,\ldots,N \tag{3.20}$$

insures that the time window constraints are included.

VII. QUALITY OF SERVICE

As above, either clock time or the number of periods may be used to represent the desired standards specified by Q_j. Hence, one could use either

$$\tau_{j+n} - \tau_j \leq Q_j \qquad \text{for every } j \in 0 \tag{3.21}$$

or

$$\sum_{i=1}^{N} \sum_{t=1}^{N} \sum_{k=1}^{K} t\, x_{i,j+n,t}^{k} - \sum_{i=1}^{N} \sum_{t=1}^{N} \sum_{k=1}^{K} t\, x_{ijt}^{k} \leq Q_j \quad ,$$

$$\text{for every } j \in 0 \tag{3.22}$$

to insure that quality of service is maintained.

VIII. OPERATIONAL CONSTRAINTS

The operational constraint may be expressed by

$$\sum_{ijt} c_{ijt}\, x_{ijt}^{k} \leq D_k \quad , \quad k = 1,2,\ldots,K \quad . \tag{3.23}$$

Although the above formulation does in fact represent the PUDP, the necessary quadruply scripted variables should be noted. The

The implication of this fact will become apparent in the next chapter where discussion of optimal solution techniques will be discussed.

CHAPTER IV

EXACT SOLUTION ALGORITHMS

It was noted in Chapter II that optimal solutions to vehicle routing problems (VRP's) were difficult to come by. Because the pickup and delivery problem (PUDP) is more complex than the VRP, one might suspect that exact solutions to it would be even more elusive. In many cases this suspicion is correct. However, for one subclass of the PUDP, large problems (over 100 stops) are readily solved optimally using a dynamic programming algorithm.

Other than dynamic programming, the exact solution techniques detailed earlier for the travelling salesman problem (TSP) and the VRP are not effective in solving the PUDP. Not all PUDP instances can be solved by dynamic programming. Consequently, this chapter is devoted to explaining why the traditionally successful algorithms fail, or are not effective when applied to the PUDP, and to developing the conditions and algorithms that allow for optimal solution to relatively large PUDP's. A detailed discussion of the multiple vehicle PUDP is deferred until Chapter VI.

I. INEFFECTIVE TECHNIQUES

Integer programming by means of cutting planes, branch and bound approaches, and Lagrangean techniques, has been used with varying degrees of success for solving the TSP and the VRP. In addition, a Benders decomposition approach has been successfully applied to the VRP

as both a heuristic and an exact solution technique. The complexity of the constraints renders each of these techniques ineffective when applied to the PUDP.

Integer Programming

The integer programming (IP) formulations of the PUDP given in Chapter III are considered as compact as possible. Although the compact formulation is only of order KN, the number of decisions variables presents the primary difficulty. At least KN^3 (0,1) decision variables are required.

Consider a hypothetical problem with 15 customers, or 31 stops. Nearly 30,000 decision variables are required for the single vehicle version of the problem and more than 100,000 variables are needed if the customers are to be serviced by four delivery vehicles. For the TSP, integer programming techniques such as cutting planes and implicit enumeration generally have performed less efficiently than have others. Given the large number of decision variables and the suspect efficiency of cutting planes in general, further pursuit of IP was not undertaken.

Branch and Bound

Branch and bound techniques are by far the most common type applied to the TSP and to the VRP. This is true because relaxations of the TSP or the VRP, such as the assignment problem, are readily solvable and provide logical branching points. Also, good lower bounds on the optimal completion of the subproblem can be readily computed.

Tight bounds and logical branching strategies are not readily available for the PUDP. The precedence requirement is the primary complicating factor. The sequence in which stops are visited is irrelevant in the TSP, but critical with the PUDP. Consequently, the relaxation solution may produce infeasible subtours. Therefore, one must not only branch to eliminate subtours, but also to achieve feasibility. Given both subtours and infeasibility, the choice of a branching rule is not readily apparent. Even if such a rule were readily available, the need to compute tight bounds remains as a necessary criterion for a good branch and bound algorithm. Such bounds are not readily available.

The PUDP is a restriction of the TSP. Therefore, the optimal solution to the TSP is a lower bound on the PUDP. For noncontrived problems, we will see that the TSP provides a very poor lower bound. Consequently, to efficiently solve the PUDP by branch and bound, a relaxation of the PUDP is needed that produces bounds better than the optimal TSP solution over the same network. Further, the bound must be easily computed in polynomial time (such a relaxation is referred to as a "polynomial time relaxation"). The following lemma shows that such a relaxation probably does not exist.

Lemma 2. Unless P = NP, there does not exist a polynomial time relaxation of the PUDP with value $\bar{R}$ which is greater than or equal to the optimal TSP relaxation solution value of Z^* for all PUDP instances.

Proof. Let P^* be the value of the optimal PUDP solution and suppose $Z^* \leq \bar{R} \leq P^*$ for every instance of the PUDP. Let A be the polynomial time algorithm that produces $\bar{R}$.

Consider the following instance of the PUDP: one vehicle, no capacity, operational, time window, or quality of service constraints other than for stop n and a requirement that all origins be visited before time T and all destinations after time T. Origin n and its corresponding destination 2n are defined such that $e_n = \ell_n = T$ and $e_{2n} = \ell_{2n} = T + 1$, $c_{n,j} = c_{1,j}$ for all j, $c_{i,n} = c_{i,1}$ for all i and $c_{n,2n} = 0$. In effect, stop n and 2n are identical to the depot.

As defined, the PUDP reduces to two TSP's. One over the n-1 origins and one over the n-1 destinations. Let $c_{ij} = 0$ for all i, $j \in D$, $c_{ij} = \infty$ for all $i \in O$ and $j \in D$. Apply A to the resulting PUDP.

Clearly P^* is the value of the optimal TSP solution over the depot and n-1 origins, which is also the value of the PUDP solution. Hence, $Z^* = \bar{R} = P^*$ from the hypothesis. But any TSP could easily be transformed to such a PUDP instance. Therefore, A can be used to solve any TSP. Since A is a polynomial time algorithm, this would imply P = NP. Consequently, we rejected $Z^* \leq \bar{R} \leq P^*$ and accept the provisions of the lemma.

Empirical evidence suggests that obtaining a bound better than the TSP bound is difficult for any instance of the PUDP. No workable bounding scheme was uncovered that was not also a lower bound for the TSP. The LP solution obtained by relaxing the integrality requirement could be such a bound, but is not easily computable due primarily to the

previously noted number of decision variables required. Given such loose bounds, any branching tree would clearly become enormous. Attempted hand solution of a five stop (two origin/destination pair) problem demonstrated the futility of the branch and bound technique. Therefore, it is not deemed practical for optimally solving the general PUDP.

Lagrangean Technique

One of the most powerful techniques for solving the symmetric TSP was the 1-tree approach of Held and Karp (22,23). The precedence requirement of the PUDP can not be accomodated by the 1-tree approach. The construction of the 1-tree requires the construction of a minimum spanning tree. The greedy algorithms used for the solution of the minimum spanning tree problem requires an independence relationship that is immediately contradicted by the "origin before destination" constraint.

The result is similar to that cited above. Whereas the TSP is sequence independent, the PUDP is sequence dependent. This dependence is the primary factor precluding or limiting the solution of the PUDP by the 1-tree, branch and bound, and integer programming approaches.

Benders Decomposition

The Fisher-Jaikamur algorithm is based on iterating between a generalized assignment problem (GAP) and a travelling salesman sub-problem. The objective of the GAP is to optimally assign customers to vehicles such that vehicle capacity is not exceeded.

With the PUDP, vehicle capacity may not even be a factor. Even if it were, the capacity limitation affects the problem differently. It is dependent upon the specific routes followed. Remaining unused vehicle capacity decreases as an origin is visited and increases as a destination is visited. Thus, for a given combination of customers, one or more sequences might cause the capacity limitation to be violated, while another would be feasible. This can be contrasted to the VRP where one normally considers the vehicle fully loaded when it departs the depot. In this varying capacity environment, the GAP is not clearly defined. Assigning customers to the correct vehicle is the crux of the multiple vehicle PUDP. However, no logical formulation of the customer assignments could be found so that the Benders decomposition concept could be applied to obtain an exact solution. Chapter VI specifically addresses the multiple vehicle PUDP.

II. DYNAMIC PROGRAMMING AND PRECEDENCE CONSTRAINTS

Dynamic programming has been used less on the TSP than some other techniques. The computer storage requirements, as noted in Chapter II, are the primary difficulty in using DP to solve the TSP. Yet dynamic programming can be adapted to handle the precedence relationship of the PUDP. Further, this can be accomplished with a significant reduction in storage requirements.

The reason for the reduction is that infeasible states are not generated. For a 13 node TSP, nearly 25,000 individual storage locations

are required for the data generated by the recursion equations. For the PUDP of equal size (six customers plus the depot) and only precedence constraints binding, less than 3,000 locations are needed. The next sections explain how this reduction in storage requirements is possible. Further reductions are possible for some more heavily constrainted instances of the problem. These instances are discussed later in the chapter.

Induction Schemes

In (2.17), the basic dynamic programming recursion for the travelling salesman problem was given. The expression for the shortest partial tour from node 1 to node j that passed through

$$S_k = \{i_1, i_2, \ldots, i_{k-1}\} \tag{4.1}$$

was

$$f_k(j|S_k) = \underset{m}{\text{minimum}}\{f_{k-1}(i_m|S_k - i_m) + c_{i_m,j}\} \quad . \tag{4.2}$$

In this form, the recursion can be thought of as forward induction in that the salesman is proceeding forward from the starting point to some intermediate point in his tour. Often, dynamic programming is thought of as backward induction.

In backward induction, the recursion seeks to find the shortest partial tour from node j that passes through

$$S_k = \{i_{N+2-k}, i_{N+1-k}, \ldots, i_N\} \tag{4.3}$$

and returns to the original origin, node 1. The subscript k indicates the total number of nodes, including node j that are included in the partial tour not including the depot. As developed in Chapter III, N is the total number of distinct nodes. Mathematically, the recursive expressions are identical. The only difference is in the definition of S_k. Backward induction is used in the algorithms developed to solve the PUDP. Therefore, all explanations to follow assume the backward orientation.

State Space Generation

Only feasible state vectors need to be generated for the PUDP. Stop j of expression (4.2) will be referred to as the lead. Consequently, if j is the lead and j is one of the n origins, it must be true that

$$(j+n) \in S_k \tag{4.4}$$

if the vector is to be feasible. Similarly, if j is a destination, it must be true that

$$(j-n) \notin S_k \quad . \tag{4.5}$$

Also, if stop i is an origin, then

$$i \in S_k \Rightarrow (i+n) \in S_k \quad . \tag{4.6}$$

State Vector Representation

A binary vector representing the lead unioned with an appropriate S_k is generated and stored for each f_k value. These vectors are used to identify each S_k and to identify the optimal solution sequence. Each vector takes advantage of the way in which decimal numbers are stored by a digital computer. If i represents any one of the N different stop numbers, then the binary representation of the decimal number 2^{i-1} consists of one 1 in the appropriate location and 0's in all other positions of the word. Suppose jUS = {2,4,6} and the computer system employs an eight bit word. Figure 4 depicts the binary representation of the decimal number 42, which is obtained by

$$2^{2-1} + 2^{4-1} + 2^{6-1} = 2 + 8 + 32 = 42 \quad . \tag{4.7}$$

Consequently, the decimal number 42 corresponds to the presence of stops number 2, 4 and 6 in the state vector. It is easily seen that all possible combinations of stops have unique binary representations.

For problems where N is greater than the length, I, of an integer word on the computer system in use, the representation is separated into two (or more) vectors such that each word represents part of the total state vector. For example, suppose N = 45 and I = 32. Then for given state *, two words, REP1 (*) and REP2 (*), could be used to indicate the state condition for $f_k(^*)$. Stop numbers 1 through 31 could be tracked by REP1 (*) and 32 through 45 by REP2 (*). The concept is similarly extended to REP3, etc, as necessary for larger problems.

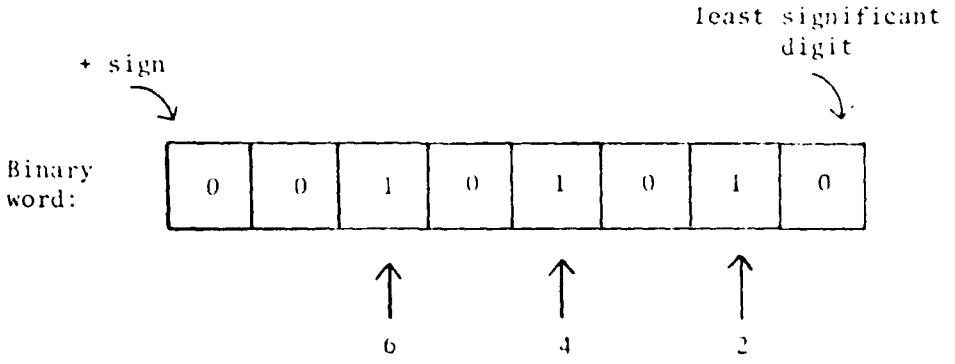


Figure 4. Binary representation of the state vector for stops 2, 4 and 6.

Storage Requirements

As noted, the number of storage locations required to solve the PUDP is less than that for the TSP. The feasibility criteria, expressed by (4.4) through (4.6), decrease the number of combinations that must be considered for each lead. This difference can best be visualized by computing the storage locations required for one step of a sample TSP and for the related PUDP.

Example requirements. Let N = 15 and k = 7. Suppose the lead is stop number 2, which corresponds to an origin in the PUDP with the corresponding destination being stop number 9. For the TSP, with the depot fixed as stop number 1, there are $\binom{13}{6} = 1,716$ different S_7 values to be computed. For the PUDP, stop number 9 must be included in all S_7 values. Thus, there are five free positions available, which can be filled with

- 2 origin/destination pairs and 1 destination,
- 1 origin/destination pair and 3 destinations, or
- 5 destinations.

The number of combinations for each configuration is

- $\binom{6}{2}\binom{4}{1} = 15 \cdot 4 = 60$,
- $\binom{6}{1}\binom{5}{3} = 6 \cdot 10 = 60$,
- $\binom{6}{5} = 6$,

respectively, for a total of only 126 values to be computed. This represents more than a tenfold decrease in storage requirements. Were

the lead position a destination, 141 storage locations would be required. For a problem with N = 29, over a hundredfold decrease can be achieved. However, the growth of storage locations remains exponential.

Total requirements. An upper limit on the total number of storage locations required of $2n \cdot 3^{n-1} + 1$ is given by Psaraftis (24), where n is the number of origin/destination pairs. His development of the states is different, but requires essentially the same number of total states for the PUDP with only precedence constraints binding. The inclusion of time window and/or quality of service constraints for some problem instances reduces storage requirements even further, in contrast to Psaraftis' development, as will be seen later in this chapter.

Example Problem

An example problem will be used to explain exactly how the dynamic programming algorithm developed solves the PUDP with only precedence constraints binding. A problem with three customers or origin/destination pairs will be assumed. The depot is node 1, while the origin/destination pairs are 2/5, 3/6, and 4/7, respectively. Table 1 depicts the cost matrix for the hypothetical problem. Inconsistent cells, such as a destination to its corresponding origin, are blacked out to so indicate. However, any cost could be placed in one of these cells, as the algorithm does not allow for such infeasibilities. Table 2 presents all of the $f_k(j|S_k)$ values, in the order generated, as well as the binary identification vectors.

TABLE 1

COST MATRIX FOR EXAMPLE DYNAMIC PROGRAMMING PROBLEM

	1	2	3	4	5	6	7
1		2	2	0			
2			5	9	8	0	3
3		5		11	0	8	11
4		4	6		3	3	0
5	0		0	8		7	8
6	2	0		8	7		0
7	1	3	11		8	0	

TABLE 2

RECURSIVE DATA FOR EXAMPLE DYNAMIC PROGRAMMING PROBLEM

	f_k VALUE	LEAD	COUNTER VECTOR	BINARY REPRESENTATION 8-7-6-5-4-3-2-1
k=2	8	2	2|5	0 0 0 1 0 0 1 0
	10	3	3|6	0 0 1 0 0 1 0 0
	1	4	4|7	0 1 0 0 1 0 0 0
	9	5	5|6	0 0 1 1 0 0 0 0
	9	5	5|7	0 1 0 1 0 0 0 0
	7	6	6|5	0 0 1 1 0 0 0 0
	1	6	6|7	0 1 1 0 0 0 0 0
	8	7	7|5	0 1 0 1 0 0 0 0
	2	7	7|6	0 1 1 0 0 0 0 0
k=3	7	2	2|56	0 0 1 1 0 0 1 0
	11	2	2|57	0 1 0 1 0 0 1 0
	9	3	3|65	0 0 1 1 0 1 0 0
	9	3	3|67	0 1 1 0 0 1 0 0
	8	4	4|75	0 1 0 1 1 0 0 0
	2	4	4|76	0 1 1 0 1 0 0 0
	10	5	5|36	0 1 0 1 1 0 0 0
	9	5	5|47	0 1 0 1 1 0 0 0
	8	5	5|67	0 1 1 1 0 0 0 0
	8	6	6|25	0 0 1 1 0 0 1 0
	9	6	6|47	0 1 1 0 1 0 0 0
	8	6	6|57	0 1 1 1 0 0 0 0
	11	7	7|25	0 1 0 0 0 0 1 0
	21	7	7|36	0 1 1 0 0 1 0 0
	7	7	7|56	0 1 1 1 0 0 0 0
k=4	14	2	2|536	0 0 1 1 0 1 1 0
	17	2	2|547	0 1 0 1 1 0 1 0
	8	2	2|567	0 1 1 1 0 0 1 0
	12	3	3|625	0 0 1 1 0 1 1 0
	13	3	3|647	0 1 1 0 1 1 0 0
	8	3	3|657	0 1 1 1 0 1 0 0
	11	4	4|725	0 1 0 1 1 0 1 0
	15	4	4|736	0 1 1 0 1 1 0 0
	7	4	4|756	0 1 1 1 1 0 0 0

TABLE 2 (continued)

	f_k VALUE	LEAD	COUNTER VECTOR	BINARY REPRESENTATION 8-7-6-5-4-3-2-1
k=4	9	5	5|367	0 1 1 1 0 1 0 0
	10	5	5|476	0 1 1 1 1 0 0 0
	11	6	6|257	0 1 1 1 0 0 1 0
	16	6	6|475	0 1 1 1 1 0 0 0
	8	7	7|256	0 1 1 1 0 0 1 0
	18	7	7|365	0 1 1 1 0 1 0 0
k=5	13	2	2|5367	0 1 1 1 0 1 1 0
	16	2	2|5476	0 1 1 1 1 0 1 0
	13	3	3|6257	0 1 1 1 0 1 1 0
	10	3	3|6475	0 1 1 1 1 1 0 0
	8	4	4|7256	0 1 1 1 1 0 1 0
	12	4	4|7365	0 1 1 1 1 1 0 0
	13	5	5|3647	0 1 1 1 1 1 0 0
	17	6	6|2547	0 1 1 1 1 0 1 0
	17	7	7|2536	0 1 1 1 0 1 1 0
k=6	15	2	2|53647	0 1 1 1 1 1 1 0
	19	3	3|62547	0 1 1 1 1 1 1 0
	17	4	4|72536	0 1 1 1 1 1 1 0
k=7	17	1	1|253647	0 1 1 1 1 1 1 1

Optimal Solutions: 1-2-3-5-4-7-6-1 and
1-4-7-2-3-5-6-1

The algorithm used actually generates a vector of length k that is incremented to obtain each feasible state, like a counter. Each element of the k-vector is then used to compute the binary representation, which is stored for each f_k value. If the lead position is an origin, the next position in the k-vector is automatically set to that origin's corresponding destination. Consequently, only k-2 positions would require incrementing. Further, any time one of the elements incremented is an origin, the next element is similarly set to its corresponding destination. Thus, for k = 5 the first counter vector is given in Table 2 as

2|5367.

Origin 2 as the lead automatically sets the next position to 2's destination, 5. Origin 3 automatically sets the position on its right to its destination, 6. Each time one element of the k-vector is incremented, all elements to the right of it are initialized to their lowest feasible node number. Then the vector is incremented like a counter, with the right most element incremented first. Once completed, each element of the k-vector is then used to compute the binary representation, which is stored for each f_k value. The binary representation conveys only the identity of the nodes, not the sequence in which they were generated.

$\underline{f_2(j|i_N)}$. The algorithm begins by computing $f_2(j|i_N)$. Recalling that the algorithm uses backward recursion, there are only two configuration patterns possible:

- an origin/destination pair with return to the depot, and
- two destinations with return to the depot.

In the latter case, any destination could serve as the lead, while in the former, only an origin could. The computation in either case is straightforward and will not be elaborated on further.

$f_3(j|S_3)$. Beginning with j, or the lead, equal to the first origin, node 2, we generate all possible S_3 combinations. Because node 2 is an origin, node 5 must be included, leaving only one free position in S_3. Clearly, this node must be a destination. Beginning with the lowest numbered destination node, not identical to any destination already fixed, each state vector is incrementally generated. Expression (4.2) and the previously generated data are then used to compute each f_3 value. Once all S_3 combinations are generated for node 2, the lead is incremented and the procedure repeated.

When the lead is incremented to node number 5, the first destination node, there are two free positions in S_3 that can be filled by:

- an origin/destination pair, or
- the remaining two destinations.

In the former case, an origin requires its corresponding destination. Thus, when the origin position in the state vector is being incremented, there is no free position left in S_3. For the latter, one free position remains.

$f_4(j|S_4)$. The procedure for f_4 values is the logical extension of that for f_3 values. With an origin as a lead, the two free positions in S_4 can be filled by either:

- an origin/destination pair, or
- two destinations.

Similarly, when the lead position is a destination, an origin/destination pair with one destination is the only possible combination. A destination followed by three destinations is impossible since there are only three total destinations in the problem.

Remaining $f_k(j|S_k)$'s. The same process is repeated for f_5 and f_6 values. For f_6 values, only origins are legitimate lead values, followed by all other node numbers. The optimal solution value is obtained by solving

$$f_7(1\ 2,3,\ldots,7) = \min_m [f_6(i_m|S_6-i_m) + c_{i_m,1}] \quad . \tag{4.8}$$

Because only origins can be in the lead position in f_7, one is assured that $i_m \in 0$ and that the route begins legitimately. One can then identify an optimal solution with value 17. One such solution is 1-2-3-5-4-7-6-1.

Obtaining the solution. Using the f_k values, the lead node for that value, the binary representation and the original cost data, this solution can be constructed one node at a time. For example, the f_7 value of 17 minus the cost from the depot to node 2, which is two, equals the f_6 value of 15 associated with node 2 as a lead. Similarly, this value of 15 minus the cost from node 2 to 3 of 5 equals the f_5 value of 10 associated with node 3 as the lead. This assignment is legitimate since the identification vectors indicate that no previously

scheduled stops (in this case node 2) were used to obtain the f_k value. The process continues until all stops are sequenced. The tour is then completed by indicating the return to the depot.

Computational Results

With only the precedence constraints binding, the exponential growth of the required storage locations limits the size of problem that can be practically solved. Table 3 summarizes results for four small problems. All computer runs were made on the University of Tennessee Computing Center's IBM 370/3031 computer system. Cost matrix data were generated randomly. Both x and y graph coordinates between 0 and 99 were generated for each point. Interpoint distances were then computed using the rectilinear metric, i.e.

$$C(i,j) = |x_i - x_j| + |y_i - y_j| \quad , \tag{4.9}$$

where x_i is the x coordinate of the i^{th} point, etc. The data generation technique was the same for all problems examined throughout this dissertation. All of the algorithms developed in this and the next two chapters are independent of the manner in which the cost matrix is obtained. The rectilinear metric was selected only becuase it facilitated hand computations during the early stages of algorithm development. Any metric could be used in place of the rectilinear one.

TABLE 3

TYPICAL COMPUTATIONAL RESULTS FOR THE DYNAMIC PROGRAMMING SOLUTION OF THE PUDP

# OF CUSTOMERS	# OF NODES	# OF $f_k(j \mid S_k)$ VALUES	SOLUTION TIME (SEC)*
2	5	10	3.92
3	7	51	4.08
5	11	805	5.23
7	15	10,199	124.72

*Includes input and output operations and data generation as well as the solution procedure.

III. DYNAMIC PROGRAMMING WITH ADDITIONAL CONSTRAINTS

The discussion in the last section dealt only with binding precedence constraints. In some real world applications of the pickup and delivery problem, many other constraints, as suggested earlier, may be present. In Chapter III, two different approaches for formulating the fully constrained version of the PUDP were presented. One employed the use of clock time, while the other used the stop number of a customer in the optimal sequence. This latter approach is ideally suited to solution by dynamic programming.

Lead Position Feasibility

Position identification. The first origin visited after the vehicle leaves the depot is considered to be the first stop number. Consequently, the last destination the vehicle visits before returning to the depot is designated stop number N-1, with all other origins and destinations being sequenced somewhere in between. Earlier in this chapter it was noted that the subscript k on the f_k indicated the number of nodes or customers in the partial tour from node j to the depot. Let τ_k represent the stop number associated with k. Then the stop number can be found by

$$\tau_k = N-k, \quad k = 2,3,\ldots,N-1 \quad . \tag{1.10}$$

Feasibility set. The dynamic programming algorithm previously described proceeded by first fixing the lead customer, and then generating all combinations of subsequent stops. However, when constraints other than precedence requirements are present, it is likely that not all customers can feasibly be assigned to a given τ_k lead position. Therefore, there is no need to generate all subsequent customer combinations, nor to store any data relative to infeasible combinations. Let

$$J_{\tau_k} = \{j | j \text{ is a feasible lead for } \tau_k\} \tag{3.11}$$

State vectors will be generated only for $j \in J_{\tau_k}$.

Time windows. In order for the complement of J_{τ_k} to not be empty for all τ_k, at least one of the customer requirements must have a not earlier than (NET) requirement or a not later than (NLT) requirement, or both. The more interesting case, in terms of solution by dynamic programming, is when each customer's requirements involve time windows, which it is assumed can be defined by stop numbers. When time windows are present for all customer requirements, the set J_{τ_k} is clearly defined for all τ_k, and the total number of storage locations required for the state and binary representation vectors is reduced. Equally significant is that many of the other possible constraints are also definable in terms of time windows.

Handling Other Constraints

The presence of NET or NLT times will be assumed for at least one portion (either the origin or the destination) of a customer's requirement. One point in time is all that is necessary to fix delivery windows, as explained below, for servicing both the customer's origin and destination. In many cases, this one point would be a NLT time for delivery to the destination. For ease of explanation, the dial-a-ride problem will be used. However, any PUDP for which the appropriate assumptions hold could be substituted equally as well.

Quality of service. Let Q represent the quality of service for the dial-a-ride service problem. Q is specified by the provider based on his standards. It represents the maximum number of stops before a given customer reaches his destination. Conversely, given a required delivery time, represented by a specific stop number, that time minus Q represents the earliest time that a customer could be picked up at his origin. The use of stop numbers to represent time is absolutely necessary. Consequently, the direct travel time between two points is taken to be one stop, regardless of the actual clock time required. Suppose a customer must be at his destination by stop T. In this case, the NLT for the customer's destination is T. The NLT time for the corresponding origin is T-1 since it takes a minimum of one period to travel from a customer's origin to his destination. Since Q is the maximum difference between when a customer must be at his destination and when a customer is actually picked up, the earliest time that the

customer could be picked up would be T-Q. Thus, T-Q represents the NET time for the customer's origin. A NET value of T-Q+1 for the customer's destination is obtained since there is a minimum of a one stop travel time. Suppose Q=5 and T=18 for a given customer. Then the NET and NLT times for that customer's destination would be 18-5+1=14 and 18. For the customer's origin, these times would be 18-5=13 and 18-1=17, respectively. The overall time window associated with this customer would thus be stop numbers 13 through 18, inclusive. In addition, there would be a window of 13 through 17 inclusive for his origin and 14-18 inclusive for his destination. Any attempt to provide service at stop numbers outside these windows would result in an infeasible solution.

If one wishes to avoid long waiting periods upon arriving at his destination, a maximum waiting parameter, M, could be used so that the NET time for arrival at the destination could be computed as T-M. Continuing the numerical example of the previous paragraph, if M=3, the earliest time (stop number) that the customer could arrive at his destination would be 18-3=15. The overall and origin time windows would remain unchanged. However, the destination time window would become stops 15 through 18 inclusive. Both Q and M were used for computational experimentation, which will be presented later in this chapter.

Capacity constraints. For the dial-a-ride problem, the capacity constraint is simpler because each person can be thought of as one unit.

Therefore, one must only be concerned that the number of people in the vehicle at any time does not exceed the vehicle's capacity. This can be accommodated by using a not unrealistic assumption that the vehicle capacity is greater than or equal to Q. Since Q represents the maximum number of stops before a customer reaches his destination, and since each customer represents a unique stop, the vehicle cpapcity can never be exceeded.

The assertion that the needed assumption is not unrealistic can be argued in terms of a hypothetical situation. Suppose our dial-a-ride service provides transportation services to handicapped people. The van to be used can hold at most seven passengers. Fifteen requests for service have been received. Each request corresponds to a pickup at an origin and delivery to a destination. Hence, our driver has 30 specific stops to make. Given that the passengers are handicapped and require assistance entering and leaving the vehicle, we suppose that the average time between any two stops in an optimal sequence is 15 minutes. When one physical location corresponds to more than one stop, the 15 minute average should not be materially affected by the increased time required to provide assistance to more than one customer. These numbers are arbitrary, but do lead to a convenient eight hour workday for the driver.

The vehicle's capacity could not be exceeded unless at least one person were required to ride for two or more hours to arrive at his destination. Note that such an eventuality is necessary but not necessarily sufficient for vehicle capacity to be exceeded. Two hours

appears excessive in terms of the quality of service that any organization providing dial-a-ride service would provide. Consequently, the assumption is judged not unrealistic.

Storage Requirements

Reduction based on lead. By using the set J_{π_k}, as defined by expression (4.11), the storage requirements for the state and representation vectors can be reduced. For example, if for each value of π_k, only 30% of the j are included in J_{π_k}, then the total storage required is only 30% of that required for the unconstrained instance. This figure makes no assumptions about the feasibility of each S_k. Further reductions are possible when this is considered.

Feasible elements in S_k. The sets S_k represent the customers who are served subsequent to the lead stop in a partial tour. For most values of k, not all stops can feasibly be included in S_k. For example, any customer j, either an origin or a destination, may have an earliest service time e_j and a latest service time ℓ_j defined. If

$$\ell_j \leq \pi_k \tag{4.12}$$

cannot feasibly be contained in S_k. Let

$$U_{\pi_k} = \{j | \ell_j \leq \pi_k\} \quad . \tag{4.13}$$

The number of f_k values is primarily determined by the number of combinations in S_k, which is determined by the number of customers to

be taken over the k-1 remaining positions. From expressions (4.12) and (4.13) it must be the case that

$$j \in U_{\pi_k} \Rightarrow j \notin S_k \tag{4.14}$$

The reduction from (4.14) in the number of customers that can be included in S_k produces a marked decrease in storage requirements. Specific examples are presented below. However, even further savings in storage and directly related computational effort can be achieved.

Required elements in S_k. While some customers can not be included in S_k, it is often true that others must be if the final tour is to be feasible, regardless of partial tour feasibility. Consider any customer j, either an origin or a destination, where

$$e_j > \pi_k \quad . \tag{4.15}$$

If customer j is not included in S_k, there can be no feasible completion of the total tour, since such a completion would require j to be sequenced before π_k, in violation of (4.15). It should be noted that the partial tour by itself might well be feasible. Let

$$V_{\pi_k} = \{j | e_j > \pi_k\} \tag{4.15.1}$$

and define

$$F_{\pi_k} = \{j | j \in ((\bar{U}_{\pi_k} \cap \bar{V}_{\pi_k}) \cup J_{\pi_k})\} \tag{4.16}$$

Then F_{π_k} represents all customers for which a decision can be made as to

whether or not they are to be included in a given partial tour. Using the symbol $|\cdot|$ to represent the set cardinality, the upper limit on the number of f_k storage locations for a given value of k is given by

$$|J_{\pi_k}| \cdot \begin{pmatrix} |F_{\pi_k}| - 1 \\ k-1- |V_{\pi_k}| \end{pmatrix} . \qquad (4.17)$$

The precedence requirements make the actual requirements significantly less.

Sample storage requirements. Consider a travelling salesman problem, a constrained and an unconstrained dial-a-ride service problem of size N=13. Suppose k=7, which represents the halfway point in the recursions. Further suppose that for the constrained problem the set J_{π_7} consists of two origins and three destinations, the set U_{π_7} consists of one origin and one destination in the form of one pair. Thus, F_{π_7} consists of three origins and four destinations. Table 4 shows the storage requirements to compute all feasible f_7 values for all three problems. The 45 actual locations required for the constrained problem is only 60% of the limit of 75 computed by expression (4.17).

TABLE 4

EXAMPLE f_k STORAGE REQUIREMENTS FOR THREE RELATED PROBLEMS WITH N=13 and k=7

Travelling Salesman Problem	Unconstrained PUDP	Constrained PUDP
5,544	576	45

As will be seen below, this efficient method of state generation and storage allows for the solution of much larger problems than previously solved.

Existence of a Feasible Solution

Feasibility question. When all of the previously suggested problem constraints are present, there is no guarantee that a feasible solution even exists. The more heavily constrained a given problem is, the greater the probability that a feasible solution does not exist. The following lemma examines the feasibility question for one instance of the PUDP where capacity constraints are not binding, or where the capacity arguments suggested earlier in this chapter are applicable.

Suppose that for each customer with origin i and corresponding destination j, all relevant constraints can be expressed in terms of the stop numbers on the vehicle's route. Further suppose that each customer has a specified stop number T_j representing the NLT time (ℓ_j) for delivery to the destination. The latest time the corresponding origin could feasibly be visited is T_j-1 since at least one time period is required to travel between any two points. Consider a solution P generated by taking each required stop in a nondecreasing ordering of the not later than times. For each customer, there are two required stops: one for the origin and one for the destination. In such an ordering, a customer's destination would always follow its corresponding origin since $\ell_j = T_j > T_j - 1 = \ell_i$. Consequently, P would always satisfy the precedence requirement of the PUDP.

The construction of P can be demonstrated by the following simple example. Suppose there are three customers with NLT times for their destinations of 4, 5 and 8 for D_1 (destination 1), D_2, and D_3, respectively. Then the NLT times for the corresponding origins are 3, 4, and 7 for O_1, O_2 and O_3. Then P would be the sequence:

$$\text{Depot-}O_1(3)\text{-}O_2(4)\text{-}D_1(4)\text{-}D_2(5)\text{-}O_3(7)\text{-}D_3(8)\text{-Depot},$$

where the numbers in parenthesis are the NLT values.

Lemma 3. Given a not later than time for each delivery, construct P as shown above. Suppose Q (the quality of service paramenter) is defined to be the maximum possible number of stops before a customer reaches his destination. Thus, for a customer with origin i, destination j and T_j, the not earlier than times are computed to be T_j-Q and T_j-Q+1 respectively (discussed earlier on p. 68). Then, there exists at least one feasible solution if and only if P is feasible. To be feasible a point must be sequenced within its time window: T_j-Q through T_j-1 inclusive for the origin associated with destination j and T_j-Q+1 through T_j inclusive for j.

Proof. The if part is obvious. For the only if part, suppose P is not feasible. Let P^* be any other sequence. It will be shown that P^* cannot be feasible. Let h represent the number of stops until the first infeasible assignment in P is encountered. Let j represent that infeasible assignment. For example, consider the following P:

Depot-O_1-O_2-D_1-O_3-D_2-D_3-Depot.

If the third stop, which is D_1, is infeasible, then h=3 and j=D_1. These are two cases to be considered, a point sequenced either too early or too late.

Case 1; $e_j > h$. In this case, the point sequenced, either an origin or a destination, has been sequenced too early. In effect, it will be shown that to move j later in the tour requires placing another point k in spot h which has $e_k \geq e_h$.

Let h' be the first stop for which j could possibly be feasibly sequenced in P^*. Clearly, $h' > h$ since $e_j > h$ in P. Hence, in P^*, h'-1 points must be sequenced before j. Let

$$H = \{i | e_i < e_j\} \quad . \tag{4.18}$$

Because

$$e_i = \ell_i - Q+1, \; i = 1,2,\ldots,N \quad , \tag{4.19}$$

it must be true that $e_i < e_j$ if and only if $\ell_i < \ell_j$. Therefore, it follows that

$$|H| < h \tag{4.20}$$

where $|\cdot|$ represent the cardinality of the set. Were this not true, j would have occupied a later position in P. For P^* to be feasible, it must be true that

$$|H| \geq h'-1 \tag{4.21}$$

i.e., there are enough points to fill the first h'-1 positions. Equation (4.21), in turn, implies

$$h > h'-1 \tag{4.22}$$

or

$$h \geq h' \tag{4.23}$$

which is a contradiction. Hence, if $e_j > h$, there is no possible feasible sequence.

Case 2; $\ell_j < h$. For the second case, the point has been sequenced too late. To be feasible, it must be sequenced earlier. However, to place j earlier in the tour, another point k which has $\ell_k \leq \ell_j$ put into position h.

Let h' be the last possible stop for which j could feasibly be sequenced in P^*, i.e., $h' = \ell_j$. Using an argument which is the reverse of Case 1, $h' < h$ and $h' - 1$ stops must be sequenced before stop h (customer j) in P^*. Let

$$H^* = \{i | \ell_i \leq \ell_j\} \quad . \tag{4.24}$$

H^* is the set of customers that must be serviced not later than ℓ_j. Since P is ordered in nondecreasing ℓ_i, it must be true that

$$|H^*| \geq h \quad , \tag{4.25}$$

which in turn implies

$$|H^*| > h' > h' - 1 \quad .$$

Consequently, P^* cannot be feasible since there are more custor must be scheduled than there are available positions. In othe if $x_j < h$, there is no possible feasible sequence.

Therefore, one concludes that if P is not feasible, no otl permutation can be feasible, and the Lemma is proved.

Feasibility in general. If the conditions of Lemma 3 do checking for feasibility is much more complicated. For exampl varies per customer requirement it may be the case that P, as developed using the procedure of Lemma 3, is infeasible, but t slight modification produces a feasible tour. For example, co the data in Table 5. An ordering P would be

$$1\text{-}0_1\text{-}0_2\text{-}D_1\text{-}0_3\text{-}D_2\text{-}D_3\text{-}1$$

which is infeasible since 0_1, the first customer's origin, is first, thus, violating $e_1 = 2$. But

$$1\text{-}0_2\text{-}0_1\text{-}D_1\text{-}0_3\text{-}D_2\text{-}D_3\text{-}1$$

is feasible. Since the dynamic programming algorithm only ge feasible state vectors, infeasibility is recognized when f_k some value of k. However, for the tour construction heuristi discussed in the next chapter, a simple technique for insurin feasibility is most important.

TABLE 5

EXAMPLE DATA FOR WHICH P DOES NOT YIELD A FEASIBLE SOLUTION

CUSTOMER	NLT DESTINATION/ORIGIN		Q
1	4	3	2
2	5	4	4
3	6	5	5

Computational Experience

Algorithm description. The algorithm used to solve the vehicle dial-a-ride service problem is an extension of that only precedence constraints were binding. The extension mer porates checks to determine which customer requirements sho and which must be considered for each value of k and choice lead position as explained above.

Types of problems run. Any number of problems could h solved. However, because the literature is void of sample which heuristics could be tested, emphasis was placed on me problems. Medium sized is defined to be 15 customers, whic 31 specific stops. Using the previously mentioned approxir minutes between stops, a convenient eight hour workday is o

Constraint definition. Both a quality of service para a maximum waiting paramenter, M, are used to determine the varying the tightness of the constraints. This impact beco significant in studying the heuristic solution procedures a seen in the next chapter. Table 6 presents the time window for one problem with N = 31, Q = 11, and M = ∞.

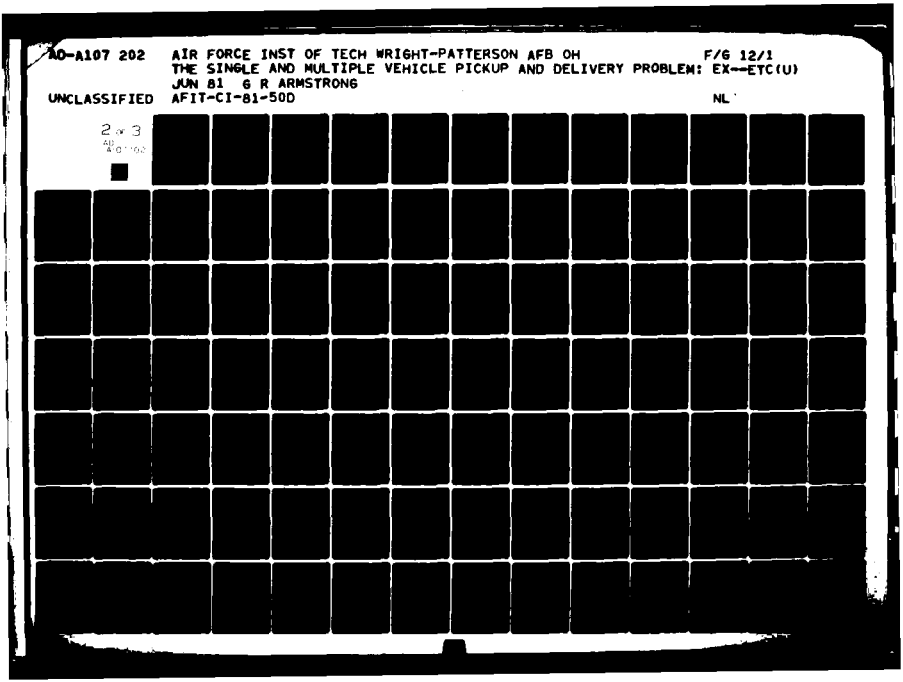
Results. Table 7 presents typical results. The stor represents the number of feasible state vectors that are g Actual storage required for N = 31 is twice the figure pre additional storage is for the representation vector. For

TABLE 6

EXAMPLE TIME WINDOWS FOR Q=11 AND M=6

#	ORIGINS NET	NLI	#	DESTINATIONS NET
2	1	3	17	2
3	1	5	18	2
4	1	6	19	2
5	1	8	20	3
6	1	10	21	5
7	3	13	22	8
8	5	15	23	10
9	7	17	24	12
10	9	19	25	14
11	12	22	26	17
12	13	25	27	18
13	16	26	28	21
14	18	28	29	23
15	20	30	30	25
16	21	31	31	26

AD-A107 202 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH F/G 12/1
THE SINGLE AND MULTIPLE VEHICLE PICKUP AND DELIVERY PROBLEM: EX--ETC(U)
JUN 81 G R ARMSTRONG
UNCLASSIFIED AFIT-CI-81-50D NL



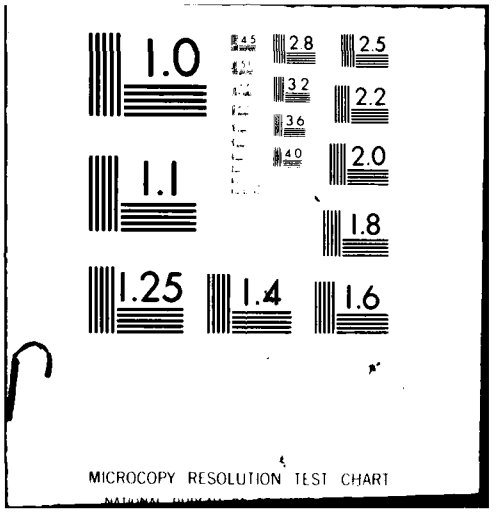

1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART

TABLE 7

SAMPLE COMPUTATIONAL REQUIREMENT FOR EXACT SOLUTION BY DYNAMIC PROGRAMMING

N	Q	M	STORAGE REQUIRED	RUN TIME (SEC)
31	5	3	174	1.280
31	5	5	227	1.433
31	7	4	453	1.706
31	11	6	784	3.037
31	11	11	867	3.623
91	7	4	1,328	1.631

total storage required is four times the figure presented. The additional storage is for the three representation vectors, one for stops 1-31, another for stops 32-62 and the last for stops 63-91. The run time is the average for 10 continuous trials.

Analysis of results. No certain explanation can be offered as to why the 91 stop problem was solved more quickly than was the similar 31 stop problem. Perhaps the answer relates to the number of other jobs in the time sharing system at the two different times these problems were solved. Notwithstanding, 91 stop problems were solved in less than two seconds, requiring allocation of less than 150k bytes of total computer storage for compilation, execution and data. One easily concludes that any practical sized single vehicle PUDP for which the simplifying assumptions — time windows and reasonable quality of service parameter — apply can be optimally solved very efficiently. The single vehicle dial-a-ride service problem appears to be an excellent practical problem for which dynamic programming could be used to obtain the exact routing provided that computer facilities were available to the agency providing the service.

IV. USES OF OPTIMAL SOLUTIONS

Despite these impressive results, dynamic programming is not practical for solving the multiple vehicle PUDP. The problem is that the number of states generated cannot be limited to the extent they can with the single vehicle case. One can no longer require customers to be

on one specific vehicle. They could just as easily be assigned to another vehicle in the optimal solution. Consequently, the cardinality of F_{π_k}, defined by expression (4.16), increases significantly, which translates into a tremendous increase in the number of f_k storage locations for a given value of k. Results pertaining to the multiple vehicle PUDP are contained in Chapter VI.

Since the PUDP has not been studied in the literature, with the exception of the few small problems Psarftis presents, there are no standard test problems in the literature against which to compare various heuristics. The dial-a-ride service problem, optimally solved by the dynamic programming algorithm, provides the means to accurately and precisely test the numerous heuristics to be discussed in the next chapter. Having the exact solution is extremely valuable, since the results indicate that many of the most popular heuristics for the TSP and the VRP perform very poorly.

V. LIMITATIONS ON DP SOLUTIONS

The results detailed above were attainable because the constraints could somehow be expressed as a function of stop numbers. This allowed for only combinations of customers in a partial tour to be considered. However, if the exact permutation of the members of a partial tour must be known to assure that constraints are not violated, then dynamic programming will not work. For example, if quantities to be moved are not unity, it is easy to create an instance where one permutation of stops is feasible, while another permutation of the same stops is

infeasible due to vehicle capacity being exceeded. The same is true of time window constraints if stop numbers cannot be assumed to be good enough approximations. Similar instances of infeasibility can be easily created for the quality of service requirement and the operational constraints. Consequently, obtaining optimal solutions appear feasible for only a rather restricted instrance of the PUDP. For the other instances heuristic solutions must suffice, since no exact solution technique has yet been found that is effective.

CHAPTER V

SINGLE VEHICLE HEURISTICS

Not all instances of the pickup and delivery problem (PUDP) appear to be able to be solved optimally. For those instances which cannot be, hueristic solutions must suffice. This necessity for reliance on heuristics was earlier noted for many instances of the travelling salesman and vehicle routing problems as well. In this chapter, therefore, heuristic solutions of the single vehicle PUDP are investigated.

I. OVERVIEW OF HEURISTIC DEVELOPMENT

Type of Instances Studied

Some instances of the PUDP can be efficiently solved optimally as shown in the last chapter. These instances require that all time related constraints be expressed in terms of stop numbers, where the stop number refers to the number of points actually visited in any partial tour. Capacity must not be a factor, or the other constraints must be such that customers do not remain on the vehicle long enough for the capacity of the vehicle to be exceeded. When these conditions hold, dynamic programming can be used to obtain the exact solution. Very large single vehicle dial-a-ride service problems can be solved when any practical quality of service parameter, Q, is used.

To say that one heuristic produces solutions that are on average so many percentage points better than another heuristic is, by itself, not too meaningful. If both heuristics produce results that are over twice

optimal, neither heuristic is very good. However, to say that a heuristic on average produces solutions within so many percentage points of optimal is meaningful. It provides a precise measure of how well a heuristic performs. For this reason, the problem instances studied in this chapter are the same as those studied in the last chapter. This allows for precise as well as relative evaluation of the heuristics.

Specific Heuristics Studied

Each of the heuristics to be discussed falls into one of the three broad classes: tour construction, tour improvement or composite. Because the Clarke-Wright savings technique dominates the commercially available vehicle routing packages, it was the logical first candidate for study. Other tour construction algorithms studied include the greedy approach and an insertion technique. The 3-optimal approach is the primary tour improvement heuristic investigated, while each of the tour construction heuristics is combined with the 3-optimal algorithm to obtain a composite solution.

Each heuristic algorithm is first discussed individually, except for the composite algorithms which require only explanation of the individual components. The algorithms developed are not straightforward extensions of those used on either the travelling salesman problem or the vehicle routing problem. As was the case with many of the exact solution techniques, the precedence relationship is the complicating factor. Computational results are finally presented which compare the performances of each of the various heuristics. The results clearly

show that tour construction heuristics do not, in general, produce good results. Again, the precedence relationship is the primary culprit.

II. CLARKE-WRIGHT HEURISTIC

Starting Point Procedures

The basic Clarke-Wright savings model was developed in Chapter II. For the pickup and delivery problem, the depot or dispatch point is taken to be the origin, and the first two points are taken to be those with the largest savings value as computed by

$$s_{ij} = c_{1i} + c_{1j} - c_{ij} \tag{5.1}$$

which can feasibly be linked together. When time windows are present, as in a dial-a-ride situation, many points cannot be linked due to the time requirements involved.

Criteria for linking. Two points can be linked only if their time windows touch or overlap. It is important to remember that all constraints are being expressed in terms of a vehicle's sequence number. Thus, an example time window of 4 through 7 inclusive for a given point, representing a customer's origin or destination, would require the vehicle to visit that point on its 4^{th}, 5^{th}, 6^{th} or 7^{th} stop after leaving the depot. Let

$$S_k = \{\text{stop numbers in k's time window}\}, \quad k = 2,3,\ldots,N. \tag{5.2}$$

Then points i and j can only be linked if the set

$$L_{ij} = S_i \cup S_j \tag{5.3}$$

contains all of the stop numbers between the lowest and the highest values. For example, suppose $S_i = \{2,3,4\}$ and $S_j = \{6,7,8\}$. Then $L_{ij} = \{2,3,4,6,7,8\}$. Because stop $5 \notin L_{ij}$, points i and j cannot be linked. If i were sequenced at its latest possible time 4, j would have a sequence time of 5 which is one stop earlier than its earliest time. Similarly, sequencing j at its earliest time of 6 would require i to be sequenced at 5, one stop beyond its latest time. Since the Clarke-Wright heuristic does not allow for the breaking of a link once formed, the linking of i to j would preclude a feasible tour from ever being constructed. Further, if

$$S_i \cap S_j = \emptyset \tag{5.4}$$

and all elements of S_i are less than those of S_j, then i must be linked to j in a forward manner, i.e.,

$$1 - i - j - 1 \tag{5.5}$$

in the initial partial tour. The same is true if the roles of i and j are reversed. Suppose $S_i = \{2,3,4\}$ and $S_j = \{5,6,7\}$. Then $S_i \cap S_j = \emptyset$. Clearly any tour having i and j feasibly linked would require i in the 4th position and j in the 5th. Any other linking would violate one of the time windows.

For other cases, one has a choice. Suppose $S_i = \{2,3,4,5,6,7\}$ and $S_j = \{4,5,6,7,8,9\}$. If i precedes j, then point i can feasibly occupy

position 3, 4, 5, 6, 7 or 8 in a complete tour. However, letting j precede i limits i to occupy only positions 5, 6, or 7. Hence, i preceding j allows greater flexibility in subsequent construction of the solution. Therefore, if $\ell_i \leq \ell_j$, then the algorithm has i preceding j in the initial partial tour.

Initial feasibility. Although the above conditions are necessary for feasibility, they are not sufficient to insure that a feasible tour can be constructed. If linking i and j precludes the subsequent construction of a feasible tour, then i and j cannot be linked. Consider a simple example, with partial data as given in Table 8. Linking i and j appears feasible with i in the stop 5 position and j in the stop 6 position. However, with i and j in these positions, point b cannot be feasibly sequenced. Yet the sequenced

$$1\text{-}x\text{-}x\text{-}x\text{-}i\text{-}b\text{-}j\text{-}x\text{-}\ldots\text{-}1 \qquad (5.6)$$

is feasible with respect to i, j and b. The x's represent arbitrary other points.

Therefore, in selecting the first two points to be linked, it is necessary to insure that a feasible final tour can be constructed. When the constraints are relatively tight, many infeasible linkings are likely to have higher savings values than those of the points that can be feasibly linked. To ignore feasibility during the construction process would almost always result in an infeasible solution. Even if one had a 90% chance of selecting a feasible linking at each stage, for

TABLE 8

EXAMPLE DATA TO DEMONSTRATE THE PROBLEMS ASSOCIATED WITH LINKING FEASIBILITY

POINT	NET	NLT
i	4	5
b	5	6
j	6	7

a problem with 15 customers (30 origin/destination points) the probability of not selecting an infeasible linking (without insuring feasibility) is less than 5%. Further, insuring a feasible linking is not sufficient to guarantee a feasible final tour as discussed above. Consequently, the probability of a feasible final tour (without insuring feasibility) is near zero.

The checking process is complicated and must be accomplished at each step before a new point is added to the solution. Assuring a feasible final tour is necessary if the heuristic is to have practical value. Infeasible solutions are considered worthless. The detailed explanation of this procedure will be deferred until later in this section.

Adding Additional Points

Recall that the Clarke-Wright algorithm adds new points, one at a time, to an existing partial tour. The new point added is always connected to the origin and one end of the partial tour. Once two points are linked, they remain linked. For ease of explanation, at any step in the process, those points which are already in the partial tour will be referred to as in the assigned set, A. The order in the assigned set will be that of the partial tour. The point being considered for addition to the partial tour is the candidate. The candidate can be inserted either in front of or behind the assigned set, provided such an insertion can lead to a feasible final tour. Therefore,

at most two feasible candidates are generated at each step of the iteration, a front end candidate and a rear end candidate. It is possible that one, but not both, of these candidates will not exist, especially near the end of the construction. The partial tour is expanded by adding to the tour the candidate which has the largest savings. In this sense, the heuristic is applied exactly in the same manner for the PUDP as it is for the TSP or the VRP.

Front end candidate. There are three possible ways that new point i could be the front end candidate, disregarding feasibility:

- i is an unassigned origin, whose corresponding destination is already sequenced.
- i is an unassigned origin as is its corresponding destination, but the assigned set is small enough that the destination can be sequenced later.
- i is an unassigned destination whose origin has not yet been sequenced.

Checking feasibility is expensive in terms of computational effort. However, simple checks can sometimes eliminate a possible candidate which passes the above criteria and has a greater savings value than the present incumbant. These checks are based on the earliest time that point i can be served (e_i) and the latest time that it can be served (ℓ_i). Let i be the candidate and j the first element of the ordered set A, which implies that i would directly precede j. If either

$$\ell_i < e_j - 1 \tag{5.7}$$

or

$$e_i \geq \ell_j \tag{5.8}$$

hold, then i is not feasible in the proposed partial tour, and can be eliminated from further consideration. Otherwise, the more complicated feasibility check is used.

Expressions (5.7) and (5.8) can be illustrated with two simple examples. First, suppose $\ell_i = 3$ and $e_j = 5$ so that (5.7) holds. If i is placed in position 3, j must be in position 4, which violates $e_j = 5$. If j holds position 5, i must be in position 4, which violates $\ell_i = 3$. Second, suppose $e_i = \ell_j = 4$. Then i in position 4 requires j in 5, violating $\ell_j = 4$, while j in position 4 requires i in 3, violating $e_i = 4$.

Rear end candidate. There are three possible ways that point i could be a rear end candidate:

- i is an unassigned destination whose corresponding origin is already sequenced.
- i is an unassigned destination whose corresponding origin has not yet been sequenced, but the assigned set is small enough that the origin could be sequenced before the first stop.
- i is an unassigned origin whose destination has not yet been sequenced.

Letting i again be the candidate and letting j be the last ordered element of A, which imples a j-i linking, i can be ignored if either

$$\ell_j + 1 < e_i \tag{5.9}$$

or

$$e_j \geq \ell_i \tag{5.10}$$

hold.

Again, two examples will illustrate why these expressions are true. First, suppose $\ell_j = 6$ and $e_i = 8$ so that (5.9) holds. If j occupies its last possible position, position 6, point i must occupy position 7. If i is in position 8, j must be in position 7. In both cases, one of the points violates a time window. For expression (5.10), suppose $e_j = \ell_i = 9$. Since j precedes i, there is no way both windows can be satisfied. The situation becomes worse as e_j increases, i.e., j cannot be served until later times.

Candidate selection. Suppose that two feasible candidates have been identified. The one with the greatest savings is added to the partial tour. If both candidates have the same savings and the front end candidate is an origin, it is sequenced. Otherwise, the rear end candidate is added to the assigned set. It is not necessarily possible to add both the front end and the rear end candidate to a partial tour in the event of a tie in savings. Final tour feasibility can be destroyed, as was discovered during initial implementation of the algorithm. In some cases the front end candidate and the rear end candidate are the same point.

Constructing a Feasible Tour

When only the precedence constraints are binding, it is a relatively simple matter to construct a feasible tour using the simple checks. However, when time windows are present, guaranteeing that a partial tour can be extended into a legitimate final solution is much more complicated.

Ordering of points. Lemma 3 of Chapter IV, establishes necessary and sufficient conditions for the existence of a feasible solution for one type of the PUDP. Although the lemma is not valid for all PUDP instances, the nondecreasing ordering of the NLT times is a useful and reasonable heuristic for testing feasibility. As before, let P denote the sequence of points ordered by nondecreasing NLT times. Therefore, in P each customer point, origin or destination, has a position number, which would correspond to the stop number if P were a proposed tour. One assumes P is feasible so that at least one feasible tour exists. If P is infeasible, the algorithm terminates.

Proposed tour Y. To test feasibility, a complete tour will be constructed using the already constructed partial tour, the candidate point, and the ordering P. The complete tour to be so constructed is termed Y. The ordering P becomes a starting point from which the proposed tour Y is created. Y is initially a null vector. If the final Y is feasible, the candidate can be feasibly linked to the partial tour. If y is not feasible, the proposed condidate is discarded; there is no assurance that a feasible tour can be constructed.

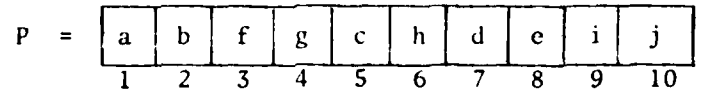
Concept of Y. Y is constructed one point at a time in reverse order. Thus, the last position is the first filled, the next to last position is the second filled, etc. The ordering P is used whenever possible. P is also examined in reverse order. Thus, the last element of P not already in Y is first considered to fill the next position. It may be that other points or the entire partial tour will be added at this point. The decision is based on assigning each point as late as feasibly possible, which in turn increases the overall likelihood of a feasible complete tour.

Latest position in A. Let L represent the position number of the point in A which has the highest position number in P. It is not necessary that the point in question be the last ordered element in A. More often than not, it will not be. Suppose there are 10 points in P and L = 7. This means that those points in positions 8, 9 and 10 of P are not yet in the Clarke-Wright generated set A. Therefore, all points in P with position numbers after L can be assigned identical positions in Y. The set A may or may not be the next points added to Y. If A contains origins whose destinations are not in A nor yet in Y, these must be inserted before A is, to preclude violation of the precedence constraints. An example test will be used to illustrate the procedure. For ease of explanation, the customer points will be assigned letters. Table 9 gives hypothetical time windows for a problem with N = 11. Figure 5 depicts the associated sequence P, a given assigned set A, and the initial assignment of stops to based on L. The fact that

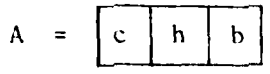
TABLE 9

HYPOTHETICAL TIME WINDOW USED TO DEMONSTRATE
TOUR CONSTRUCTION FEASIBILITY

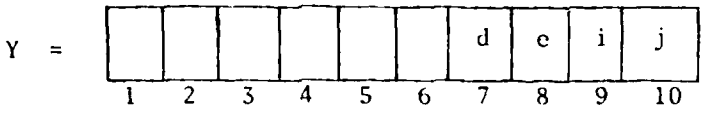
POINT #	ORIGINS NET	NLT	POINT #	DESTINATIONS NET	NLT
a	1	4	f	2	5
b	1	5	g	2	6
c	2	7	h	3	8
d	5	10	i	6	11
e	6	11	j	7	12



(a) P: the nondecreasing ordering by NLT times



(b) Given assigned set A including candidate



(c) Initial construction of Y

Figure 5. Initial construction of the proposed tour Y using P.

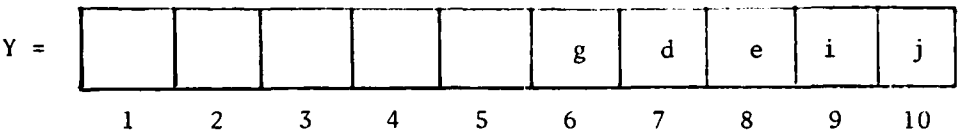
L=6 is based on the observation that point h occupies position 6 in P, whereas the other points, d, e, i and j, which correspond to stops 7, 8, 9 and 10 in P, occupy identical positions in Y.

Destination check. If the assigned set, A, contains origins whose corresponding destinations are not in A, these destinations must follow A. Further, if these destinations have position numbers in P less than L, they are not included in the already sequenced points in Y. Such stops are inserted, in nondecreasing order of NLT times, before the points in A are. In the example, destination g, corresponding to origin b, which is included in A, has a position in P of 4. Consequently, g is not yet included in Y, but must follow all elements of A. Figure 6(a) shows stop g sequenced.

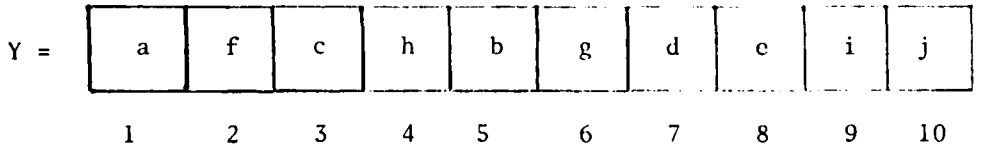
Flushing out Y. The remaining process places A directly preceding the first sequenced point in Y, and fills out by taking the as yet unsequenced stops in the same order as in P. Figure 6(b) depicts the final proposed tour, which is then checked for feasibility. The example data results in a proposed tour Y which is feasible. Therefore, the candidate, either c or b, can be added to the partial tour, and will be if it subsequently represents the greatest savings.

Clarke-Wright Solution

This process is continued until the final tour is constructed. Final tour feasibility is assured because a candidate is added to any partial tour only if a feasible final tour has actually been verified.



(a) Y with point g added to previous Y



(b) Final proposed tour

Figure 6. Continued construction of the proposed tour Y using P.

The final Clarke-Wright solution was also used as an input to the 3-optimal algorithm, thus forming a composite heuristic. Both the 3-optimal algorithm and results of the computational experiments are presented later in this chapter.

III. ROUTE INSERTION HEURISTIC

Description of Concept

The route insertion heuristic combines the philosophy of the insertion heuristics with the savings concept of the Clarke-Wright heuristics. One of the explanations offered for why the Clarke-Wright heuristic works so well on the travelling salesman problem (TSP) and vehicle routing problem (VRP) is that the more distant customers are considered early for inclusion in the tour. This procedure precludes expensive, last minute adjustments to sequence these customers. The route insertion procedure will attempt to incorporate this feature of the Clarke-Wright model.

The insertion procedures for the TSP and the VRP call for the selection of the next single point to be added to a partial tour based on some selection criteria. Because of the precedence relationship of the PUDP, origin/destination pairs, rather than a single point, are selected for insertion into the partial tour. In this manner, the route insertion heuristic specifically addresses the precedence requirement. Further, when an origin is sequenced, the vehicle must subsequently travel to the corresponding destination. Therefore, considering the locations of both the origin and the destination, relative to a partial tour, seems both logical and advantageous.

Selection Criteria

Pairs are considered for insertion in nonincreasing order of individual service cost. Individual service means a pair is serviced directly from the depot and the vehicle returns to the depot upon completion. For origin i and corresponding destination j, the individual service cost would be

$$c_{1i} + c_{ij} + c_{j1} \quad . \tag{5.11}$$

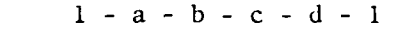
The pair with the highest individual service cost starts the process. The highest cost pair is taken first so that the most difficult or costly customers to serve are considered first, just as they were in the basic Clarke-Wright model. Thus, if pair (i,j) has the greatest cost, the initial partial tour would be

$$1 - i - j - 1 \tag{5.12}$$

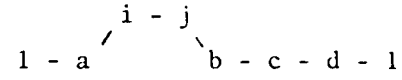
where, as before, the depot is designated as the first point. The pair with the greatest remaining cost is the next pair to be inserted into the partial tour, and so on.

Insertion Criteria

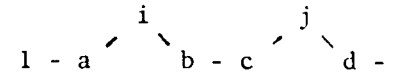
The next pair is inserted into a partial tour so as to minimize the total increase in cost. Two cost formulas apply depending on whether the new destination is inserted directly after its origin, or at a point later in the partial tour. Figure 7 (a) depicts a partial tour. Pair (i,j) is to be inserted into this partial tour. Figure 7 (b)



(a) Partial tour



(b) One insertion of pair (i,j)



(c) Another insertion of pair (i,j)

Figure 7. Example insertion patterns for the pair insertion heuristic.

depicts one possible arrangement in which j directly follows i. The increase in cost is

$$c_{ai} + c_{ij} + c_{jb} - c_{ab} \quad . \tag{5.13}$$

Figure 7 (c) depicts the more common occurrence where there are intervening stops between i and j. For this configuration the increase in cost is

$$c_{ai} + c_{ib} - c_{ab} + c_{cj} + c_{jd} - c_{cd} \quad . \tag{5.14}$$

In either case, the insertion of i and j must be such that a feasible final tour can be constructed. Insuring that a feasible final tour can be constructed from a partial tour is absolutely necessary, as noted above in the discussion of the Clarke-Wright heuristic. A somewhat similar procedure is used to verify the existence of such a feasible tour.

Feasibility of Insertion

Rough checks. If point y is inserted directly after point x, point x will precede y in all subsequent tours, although x will not directly precede y if other stops are inserted between them. Let (i,j) be the pair to be inserted and let a-b be the two points between which i is to be considered for insertion. Then i cannot be feasibly inserted between a and b if either

$$c_a \geq \ell_i \quad , \tag{5.15}$$

or

$$e_i \geq \ell_b \quad . \tag{5.16}$$

Both represent configurations that cannot possibly be feasible. Similarly, let c-d be the two points between which j is to be considered for insertion. Then j cannot be feasibly inserted if either of the following hold:

$$e_c \geq \ell_j \quad , \tag{5.17}$$

or

$$e_j \geq \ell_d \quad . \tag{5.18}$$

More complicated check. When the rough check indicates that a given insertion may be feasible, a somewhat more complicated check can be used. This check insures relative feasibility among the elements of the partial tour which includes the pair currently being inserted. Let k represent the difference between the position arbitrary point b occupies in the partial tour and the position point a occupies. If

$$e_a + k > \ell_b \tag{5.19}$$

for any a and b in the partial tour where a precedes b then the partial tour cannot be feasibly extended.

Detailed Feasibility Check

When a proposed insertion passes all of the above checks it is still possible that a feasible tour cannot be constructed. Therefore, a complete tour is generated using the ordering P discussed earlier. If this tour is feasible, the proposed insertion positions replace, or become, the incumbent positions. In either case, the next set of positions are then checked and the process repeated until the feasible positions that minimize either (5.13) or (5.14) are found.

Difference in concept. The primary difference between checking feasibility with route insertion and with the Clarke-Wright approach is that with route insertion only the order within the partial tour A must be preserved. With Clarke-Wright, the entire set A was sequenced as a block.

Construction of Y. A complete tour Y is constructed and then checked to see if it is feasible. Y is constructed one point at a time in reverse order using P and the ordered set A. Elements from set A are sequenced in Y when they are the last element and normally appear in the reverse sequence of P or when the last element of A must be sequenced to preclude violation of a "not earlier than" requirement.

Modification of e_j. When points appear in a partial tour, the earliest time one stop can actually be visited may be altered by a previous point. For example, consider the partial tour a-b-c-d, and suppose e_a is greater than any of the other times. Recall that all time

windows are defined by the stop numbers of the vehicles route. Thus, direct travel between two points requires one stop. Then the earliest that b, c and d can be visited is e_a+1, e_a+2, and e_a+3 respectively. Were any of these points visited earlier, e_a would be violated. For example, if a tour were being constructed in reverse sequence, stop d must be sequenced by the e_a+3 position (appear in the tour at e_a+3 or later) or e_a will be violated. The concept of adjusting e_j's is used in constructing Y.

Example Y. Consider the example P given in Figure 5(a), on p. 99, and time windows defined by the data in Table 8, on p. 91. Suppose that the partial tour, made up of two pairs, is

$$1 - b - d - g - i - 1 \quad . \tag{5.20}$$

Table 10 gives the revised not-earlier-than times for each of the points in the partial tour. Note the times have changed for points g and i. Figure 8 gives the step by step construction of Y. Steps 1 insert the last three elements of P. At step 4, d is not added to Y because g, not d, is on the end of the not-yet-sequenced points in A. Therefore, h, the point preceding d in P, is inserted at this step. At step 5 and 6, g and d are inserted because of their adjusted not-earlier-than times. The remaining points are inserted based on their position in P. The final Y is seen to be feasible. A forward construction using adjusted not-later-than times could also have been used.

TABLE 10

REVISED NOT EARLIER THAN TIMES FOR THE POINTS
IN A FOR EXAMPLE PROBLEM

POINT #	NET
b	1
d	5
g	6
i	7

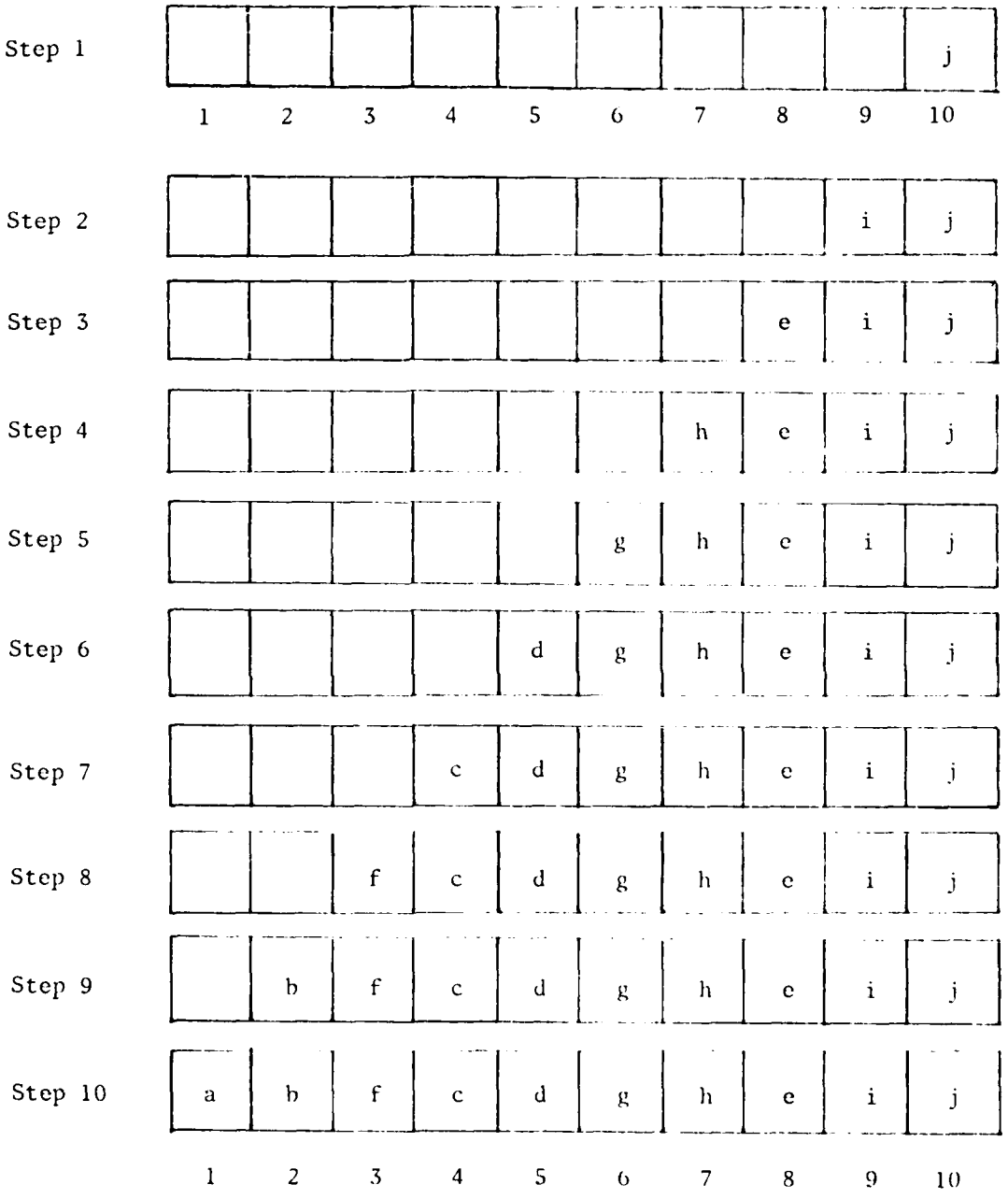


Figure 8. Step by step construction of Y.

IV. GREEDY HEURISTIC

The "greedy" or "nearest neighbor" heuristic is significantly simpler to implement than either the Clarke-Wright or the route insertion algorithms. One begins at the depot and always proceeds to the closest point which can lead to a feasible completion.

Candidate Set

At each step, a candidate set is identified. Initially, the candidate set is made up of all origins that can be visited during the first time period, i.e., all

$$j \in 0 \ni e_j < 2 \quad . \tag{5.21}$$

The closest origin is added to the tour. Its destination, if feasible, and all

$$j \in 0 \ni e_j < 3 \tag{5.22}$$

are added to the candidate set for the next stop. So long as a feasible complete tour can be attained, this process continues until all customers are scheduled. The candidate set at each point consists of unsequenced origins and of unsequenced destinations whose corresponding origins are sequenced, both of whose time windows overlap the next stop number. Arbitrary selection of the nearest point among this set will not necessarily lead to a feasible tour.

Feasibility Check

The nearest neighbor is selected from the candidate set and a trial complete tour Y is constructed. If Y is feasible, the point selected is added to the tour. If Y is not feasible, the point is removed from the candidate set and the process repeated until a feasible point is found. As before, the sequence P is used, this time in a forward manner. The k stops selected by the greedy procedure become the first k stops in Y. All remaining stops are then sequenced using the same relative order as these stops have in P.

Random Feasible Tours

The candidate set and feasibility check used for the greedy heuristic were also used with a random selection of the elements from the candidate set. These tours were used as a random input to the 3-optimal heuristic. The greedy as well as the randomly generated tour results will be discussed after the tour improvement heuristics are examined.

V. TOUR IMPROVEMENT HEURISTICS

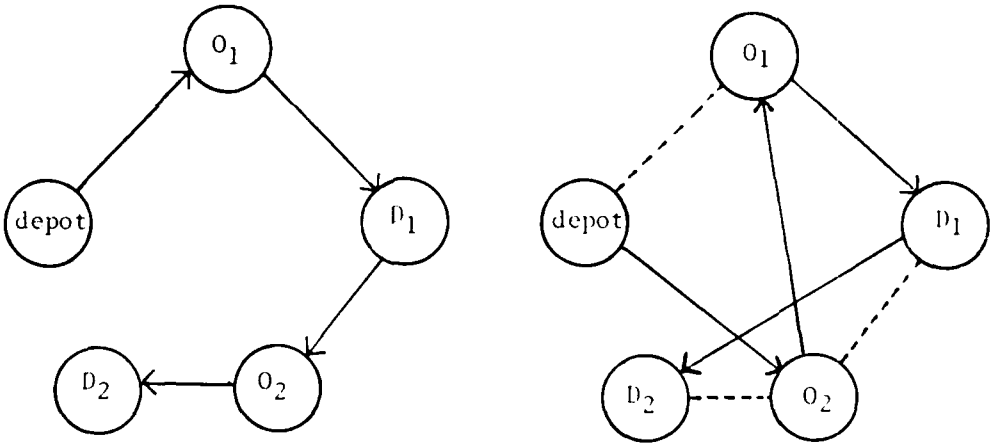
λ-Optimality

The λ-optimal heuristic of Lin and Kerninghan, discussed in Chapter II, is one of the most powerful heuristics available for use on the travelling salesman problem and the vehicle routing problem. One of the keys to the computational efficiency of the algorithm is that profitable reconnection patterns are pursued only if it is possible to

actually exchange the λ arcs and obtain a feasible solution. Insuring feasibility is relatively easy for the TSP, but much more complicated for the PUDP.

Consider the example depicted in Figure 9(a). There is no possible way to remove two arcs and reconnect them in a new pattern such that a new tour satisfies the precedence requirements. Yet Figure 9(b) shows a feasible 3-optimal reconnection. The λ-optimal approach, which only continues if the arcs already removed can be feasibly rejoined, would have stopped at $\lambda = 2$ and never have investigated the configuration shown in Figure 9(b).

One should note that for a directed TSP tour, no 2-arc exchanges are possible. A directed TSP is a TSP on a directed network, i.e., arcs only go in one direction between nodes. The 2-arc reconnection pattern, shown earlier in Figure 9, requires one of the paths remaining after the arcs are removed to be traversed in the reverse direction. Since reverse arcs do not exist in the directed network, no 2-arc exchanges are possible. Two-arc exchanges are quite likely in the PUDP. Traversing paths in a reverse direction is not prohibited in the PUDP unless doing so violates a precedence constraint or one of the time window constraints. Traversing paths in the reverse direction is discussed in greater detail later in this chapter as part of the discussion of an r-optimal algorithm. The implication of this discussion is that there is no way of knowing a priori whether or not the removal of more arcs will eventually produce feasibility. The situation depicted in Figure 9(a) is a fairly common occurrence.



(a) No 2 arc exchange possible. (b) A feasible 3 arc exchange.

Figure 9. The problem with precedence constraints and λ-optimality.

When the problem constraints are relatively tight, many reconnections will appear profitable from a cost savings point of view. Therefore, the alternative of continuing until no more savings are possible and then testing feasibility would be both inefficient and ineffective. No other alternative was found that allowed for modification of the λ-optimality concept so that it could be applied to the PUDP. This is yet another example of how the precedence constraints of the PUDP complicate or relegate ineffective solution techniques that are very powerful when applied to the TSP or the VRP.

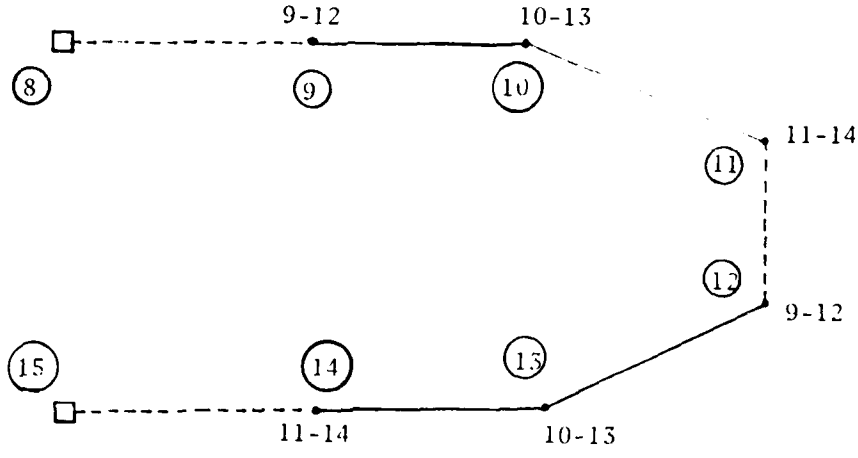
r-Optimality

The concept underlying the r-optimal heuristic was presented in Chapter II. A 3-optimal algorithm was developed and used to solve the PUDP. The 3-optimal heuristic was used on randomly generated feasible tours as well as in conjunction with solutions generated by the tour construction heuristics.

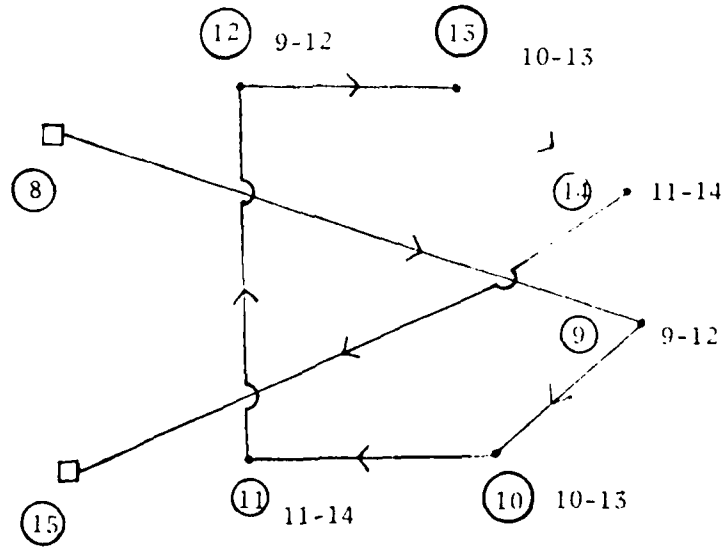
Order of complexity. For the TSP, the r-optimal heuristic is of order N^r, where N is the number of problem nodes. For the PUDP with a fixed quality of service parameter Q, the algorithm can effectively be used in order NQ^{r-1}. Recall that Q is the maximum difference between pickup and latest delivery for a customer. The reason for this is that it is highly improbable that a feasible reconnection pattern exists when the difference between the first arc removed and the last arc removed is greater than or equal to Q arcs. In fact, in all of the problems run, not once was a better solution obtained by considering more than a span of Q arcs.

Figure 10 depicts a span of Q arcs using a hypothetical tour. In this figure, letters represent the points visited and numbers the connecting arcs. Suppose Q=5 and arc 3 is the first arc removed. Then the remaining two arcs can be selected from among arcs 4 through 7. The question of feasibility provides an insight as to why such a reduction in computational effort is possible. Notwithstanding, Figure 11 shows a contrived example where a span of 2Q-1 arcs results in a feasible reconnection pattern. In the figure, Q=4; the points represent either all origins or all destinations; the squares the beginning and ending points for the six arcs in the span; the numbers inside circles the position number for a point in the tour; and the paired numbers the applicable time windows. Each point represents either an origin or a destination. Consequently, the time windows contain Q stops, not the Q+1 stops which define the time window for the pair. This is true because Q is effectively the difference between the latest delivery and the earliest pickup for a given customer.

Feasibility and reverse tours. Figure 12 is a copy of Figure 3, p. 28, previously discussed in Chapter II. Note that only in reconnection pattern (2) are all paths (dark lines remaining after the arcs are removed) traversed in the same forward direction as is the original, feasible tour. In all others, at least one sequence of stops is traversed in the reverse order. In patterns (1) and (5), both paths are traversed in reverse order. Because of the precedence requirements of the PUDP, reverse paths place very stringent requirements on the



(a) Initial tour with 3 arcs removed



(b) New tour after reconnection

Figure 10. Example of a 2Q-1 feasible reconnection pattern for Q=4.

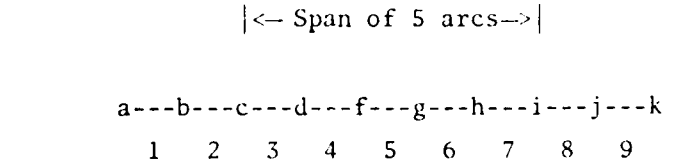
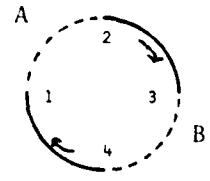
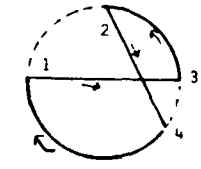


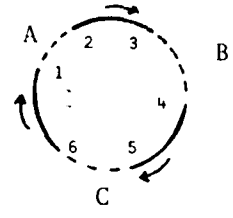
Figure 11. A span of Q=5 for the 3-optimal heuristic.

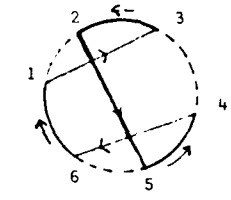




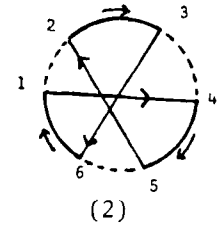
(1)

(a) 2-optimal

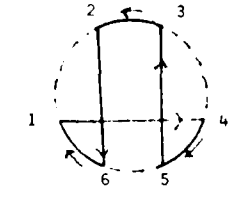




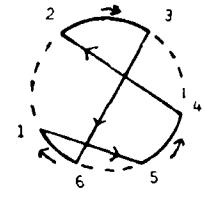
(1)



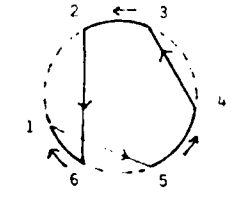
(2)



(3)



(4)



(5)

(b) 3-optimal

Figure 12. Reconnection patterns to explain why a span of Q arcs is sufficient.

points that can be legitimately included in these paths. Any set of points to be traversed in reverse order cannot feasibly contain an origin/destination pair. Also, interchanging the relative order that the disconnected paths are to be traversed restricts the permissable location of a corresponding destination. For example, in patterns (4) and (5), any origin among the points between 4 and 5 must have its destination at or beyond point 6.

Feasibility and timing. Because of the assumed time windows associated with each customer point, many reconnections result in one or more time windows being violated. For example, if any of the stops from points 2 to 3 are already sequenced at their not-later-than times, patterns (2), (3), (4) and (5) will all result in that stop being visited at a later time. For noncontrived problems, many solution points are sequenced at or near one end or the other of their respective time windows. Consequently, slight shifts, imposed by the various reconnection patterns, can result in a violation of the time window. For this reason, a span of Q stops was sufficient to obtain the best 3-optimal tour with a very high probability.

Checking feasibility. A proposed tour Y is constructed and then checked for feasibility. Because any tour presented to the 3-optimal algorithm is required to be feasible, it is only necessary to check that portion of Y between points 1 and 6 to determine if the entire tour if feasible. The check insures that the origin precedes its corresponding destination and that all time window constraints are

satisfied. When a feasible improvement is found, this solution replaces the incumbent and the algorithm is repeated. When no improvement can be found for an entire tour, the current solution is the 3-optimal solution. Despite the restrictions imposed on the reconnection patterns by the precedence and time window constraints, the 3-optimal algorithm was clearly superior to all other heuristics tested.

VI. COMPUTATIONAL RESULTS

All of the heuristics discussed above were tested using randomly generated data. In addition, the ordering P was used as an initial feasible solution for the 3-optimal algorithm. Therefore, the 3-optimal approach was used both as a tour improvement heuristic and in conjunction with one of the tour construction heuristics as a composite heuristic.

Heuristics Verses Optimality

Data. When time windows are present, as in the dial-a-ride service problem, optimal solutions are readily available which can be used to compare various heuristic's effectiveness. Table 6, on p. 81, contains the not-later-than times used for defining the time windows. Table 11 presents the solution values for 10 problem instances. The instances are different in that the location of each point, either an origin point or a destination point, is randomly generated. Consequently this results in entirely different cost matrices. Costs are again computed using the rectilinear metric. The data in Table 11 assume a quality of

TABLE 11

HEURISTIC VERSES OPTIMAL SOLUTION VALUES ON TEN
RANDOM PROBLEM INSTANCES, N=31, Q=5, M=5

PROBLEM NUMBER	OPTIMAL	CLARKE-WRIGHT	ROUTE INSERTION	NEAREST NEIGHBOR	CLARKE-WRIGHT PLUS 3-OPTIMAL	ROUTE INSERTION PLUS 3-OPTIMAL	NEAREST NEIGHBOR PLUS 3-OPTIMAL	3-OPTIMAL ON ORDERING P	BEST OF 5 3-OPTIMAL RANDOM TOUR
1	1,684	1,888	1,824	1,684	1,746	1,712	1,684	1,732	1,712
2	1,582	1,640	1,618	1,920	1,586	1,618	1,600	1,618	1,586
3	1,228	1,566	1,320	1,394	1,256	1,298	1,256	1,348	1,256
4	1,622	1,862	1,876	1,726	1,676	1,730	1,676	1,622	1,622
5	1,624	2,082	1,744	1,816	1,624	1,744	1,712	1,644	1,624
6	1,600	1,726	1,744	1,874	1,650	1,620	1,632	1,638	1,628
7	1,548	1,762	1,702	1,672	1,562	1,624	1,640	1,548	1,548
8	1,646	2,172	1,904	1,872	1,822	1,818	1,782	1,692	1,686
9	1,456	1,654	1,482	1,636	1,476	1,456	1,476	1,456	1,456
10	1,392	1,506	1,640	1,436	1,404	1,424	1,404	1,454	1,406
TOTAL	15,382	17,853	16,854	17,030	15,802	16,044	15,862	15,732	15,524
HEURISTIC TOTAL / OPTIMAL TOTAL		1.161	1.096	1.107	1.027	1.043	1.031	1.023	1.009

service, Q, of five stops and no restriction on maximum waiting time, M. Later, varying the quality of service parameters is examined.

Tour construction comparison. The data indicate that none of the tour construction heuristics perform particularly well. While the average performance on the ten problems is in the neighborhood of 12% over optimal, there are several cases where solutions of 20% or more above optimal are obtained. Clearly, there is nothing to indicate that any of the tour constuction heuristics is superior to another. These results are interesting in that the Clarke-Wright heuristic, which has shown such considerable success on the related TSP and VRP, performs so poorly on the more heavily constrained PUDP. Also noteworthy is that the fairly simple greedy procedure produces results that are on an average as good as the other more complicated tour construction heuristics. An explanation, which again related to the precedence relationship, is offered beginning on page 129.

Tour improvement and composite heuristics. The 3-optimal heuristic applied to the ordering P unquestionably produces better results than any of the tour construction heuristics. In fact, these results are equal to those of the composite heuristics. On average, the 3-optimal procedure on any given feasible tour produces results within 3% of the optimal solution, although some of the 3-optimal solutions are more than 10% above optimal. When the best of five 3-optimal solutions is used the results are on average within 1% of optimal. The five initial feasible tours are randomly generated. These results are considered

excellent, but the computational effort is five times as great as finding the 3-optimal solution from the ordering P. There does not appear to be any rationale for using a composite heuristic. The expense of finding the initial feasible tour by one of the tour construction heuristics is too great, and offers no apparent advantage over the simpler techniques.

Varying Service Parameters

The results discussed above assumed a quality of service of five stops and no special restriction on early arrivals (Q=M=5). The question next to be addressed is how altering these parameters affects the efficiency of the heuristics. Tables 12 thru 15 present solution data for: (1) Q=M=11; (2) Q=11, M=6; (3) Q=7, M=4; and, (4) Q=5, M=3 respectively. The cost data and not-later-than times for the destinations are identical to that used to obtain the data discussed above. Therefore, all solution differences are directly attributable to changes in the service parameters.

Comments on results. As the constraints become less binding, the tour construction heuristics perform even more poorly. Solutions of more than 50% above optimal were obtained. The 3-optimal heuristic also showed some deterioration in effectiveness as the constraints were relaxed. However, the average results were still within 5% of the optimal. As before, there is no advantage to using any of the tour construction heuristics in a composite mode. The 3-optimal algorithm applied to any feasible tour performs as well as any of the composites.

TABLE 12

HEURISTIC VERSES OPTIMAL SOLUTION VALUES ON TEN RANDOM PROBLEM INSTANCES, N=31, Q=11, M=11

PROBLEM NUMBER	OPTIMAL	CLARKE-WRIGHT	ROUTE INSERTION	NEAREST NEIGHBOR	CLARKE-WRIGHT PLUS 3-OPTIMAL	ROUTE INSERTION PLUS 3-OPTIMAL	NEAREST NEIGHBOR PLUS 3-OPTIMAL	3-OPTIMAL ON ORDERING P	BEST OF 5 3-OPTIMAL RANDOM TOUR
1	1,368	1,482	1,712	1,692	1,388	1,586	1,418	1,466	1,370
2	1,482	1,716	1,874	1,724	1,508	1,570	1,650	1,536	1,496
3	1,102	1,532	1,670	1,584	1,142	1,144	1,222	1,266	1,144
4	1,474	1,720	2,000	1,882	1,556	1,558	1,586	1,474	1,592
5	1,420	1,928	1,770	1,660	1,514	1,674	1,566	1,464	1,438
6	1,466	1,578	1,962	1,622	1,496	1,508	1,556	1,522	1,482
7	1,264	1,538	1,342	1,444	1,340	1,324	1,314	1,306	1,264
8	1,372	2,138	1,686	1,810	1,372	1,488	1,440	1,544	1,440
9	1,238	1,666	2,030	1,542	1,446	1,592	1,320	1,330	1,238
10	1,180	1,434	1,604	1,548	1,274	1,286	1,280	1,300	1,180
TOTAL	13,366	16,732	17,650	16,508	14,036	14,730	14,352	14,208	13,644
HEURISTIC TOTAL / OPTIMAL TOTAL		1.252	1.320	1.235	1.050	1.102	1.074	1.063	1.021

TABLE 13

HEURISTIC VERSES OPTIMAL SOLUTION VALUES ON TEN
RANDOM PROBLEM INSTANCES, N=31, Q=11, M=6

PROBLEM NUMBER	OPTIMAL	CLARKE-WRIGHT	ROUTE INSERTION	NEAREST NEIGHBOR	CLARKE-WRIGHT PLUS 3-OPTIMAL	ROUTE INSERTION PLUS 3-OPTIMAL	NEAREST NEIGHBOR PLUS 3-OPTIMAL	3-OPTIMAL ON ORDERING P	BEST OF 5 3-OPTIMAL RANDOM TOUR
1	1,368	1,482	1,772	1,692	1,388	1,502	1,418	1,466	1,396
2	1,482	1,716	1,902	1,724	1,508	1,534	1,650	1,536	1,536
3	1,102	1,532	1,670	1,584	1,142	1,144	1,222	1,266	1,142
4	1,474	1,720	2,000	1,882	1,556	1,558	1,586	1,474	1,510
5	1,428	1,928	1,762	1,660	1,514	1,682	1,566	1,472	1,472
6	1,466	1,578	1,962	1,872	1,496	1,508	1,510	1,522	1,510
7	1,264	1,538	1,342	1,444	1,340	1,324	1,314	1,306	1,374
8	1,372	2,176	1,686	1,596	1,450	1,488	1,440	1,544	1,440
9	1,238	1,666	1,968	1,542	1,446	1,522	1,320	1,330	1,462
10	1,180	1,434	1,552	1,390	1,274	1,242	1,190	1,300	1,218
TOTAL	13,374	16,770	17,616	16,386	14,114	14,504	14,216	14,216	14,060
HEURISTIC TOTAL / OPTIMAL TOTAL		1.254	1.317	1.225	1.055	1.084	1.063	1.063	1.051

TABLE 14

HEURISTIC VERSES OPTIMAL SOLUTION VALUES ON TEN
RANDOM PROBLEM INSTANCES, N=31, Q=7, M=4

PROBLEM NUMBER	OPTIMAL	CLARKE-WRIGHT	ROUTE INSERTION	NEAREST NEIGHBOR	CLARKE-WRIGHT PLUS 3-OPTIMAL	ROUTE INSERTION PLUS 3-OPTIMAL	NEAREST NEIGHBOR PLUS 3-OPTIMAL	3-OPTIMAL ON ORDERING P	BEST OF 5 3-OPTIMAL RANDOM TOUR
1	1,526	1,724	1,532	1,698	1,596	1,532	1,632	1,562	1,528
2	1,524	1,716	1,662	1,716	1,634	1,576	1,586	1,576	1,524
3	1,102	1,556	1,288	1,422	1,102	1,144	1,204	1,296	1,102
4	1,484	2,090	1,614	1,626	1,772	1,512	1,588	1,626	1,538
5	1,476	1,994	1,710	1,666	1,518	1,690	1,540	1,514	1,498
6	1,474	1,730	1,764	1,766	1,520	1,486	1,520	1,538	1,492
7	1,384	1,538	1,678	1,626	1,390	1,508	1,440	1,508	1,440
8	1,496	2,076	1,576	1,514	1,632	1,496	1,514	1,496	1,496
9	1,238	1,698	1,498	1,440	1,364	1,330	1,238	1,330	1,238
10	1,266	1,434	1,714	1,396	1,332	1,362	1,316	1,362	1,268
TOTAL	13,970	17,556	16,036	15,870	14,860	14,636	14,578	14,808	14,124
HEURISTIC TOTAL / OPTIMAL TOTAL		1.255	1.148	1,136	1.064	1.048	1.044	1.060	1.011

TABLE 15

HEURISTIC VERSES OPTIMAL SOLUTION VALUES ON TEN
RANDOM PROBLEM INSTANCES, N=31, Q=5, M=3

PROBLEM NUMBER	OPTIMAL	CLARKE-WRIGHT	ROUTE INSERTION	NEAREST NEIGHBOR	CLARKE-WRIGHT PLUS 3-OPTIMAL	ROUTE INSERTION PLUS 3-OPTIMAL	NEAREST NEIGHBOR PLUS 3-OPTIMAL	3-OPTIMAL ON ORDERING P	BEST OF 5 3-OPTIMAL RANDOM TOUR
1	1,684	1,852	1,824	1,684	1,746	1,712	1,684	1,732	1,712
2	1,594	1,640	1,648	1,752	1,594	1,594	1,608	1,622	1,608
3	1,258	1,578	1,336	1,410	1,274	1,314	1,274	1,364	1,314
4	1,622	1,852	1,780	1,726	1,676	1,622	1,676	1,622	1,622
5	1,644	2,020	1,826	1,868	1,644	1,644	1,746	1,644	1,644
6	1,604	1,682	1,692	1,740	1,654	1,624	1,636	1,642	1,624
7	1,604	1,762	1,642	1,714	1,618	1,618	1,682	1,604	1,604
8	1,698	2,154	1,904	1,872	1,822	1,818	1,782	1,744	1,716
9	1,506	1,622	1,532	1,692	1,518	1,506	1,518	1,542	1,506
10	1,402	1,494	1,452	1,424	1,404	1,454	1,404	1,434	1,404
TOTAL	15,616	17,616	16,636	16,882	15,950	15,886	16,010	15,950	15,754
HEURISTIC TOTAL / OPTIMAL TOTAL		1.128	1.065	1.081	1.021	1.017	1.025	1.021	1.009

Large Problems

It was anticipated that the basic results discussed above would carry over to large scale problems. Table 16 verifies this fact for three problems with N=91, Q=7 and M=4. The best of the five 3-optimal solutions again produced excellent results that were within 2% of the optimal value.

Analysis of Results

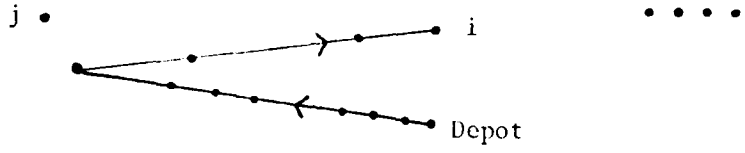
The precedence requirement is deemed to be the primary culprit which renders the tour construction heuristics ineffective. Because each of these heuristics constructs tours which maintain either actual or relative order, the order becomes locked in. Not only is the point being sequenced locked into a given relative order, its corresponding pair-mate is also. Only in the pair selection heuristic is the cost of sequencing the pair-mate considered at all when attempting to sequence a given point. However, based on the empirical data, even explicitly considering such costs is not necessarily effective.

Figure 13 depicts an example of how the precedence requirement might lead to problems with the Clarke-Wright or greedy heuristics. In panel (a) one supposes that a partial tour, depicted by the dark line, has been constructed and origin i is determined to be the next point to be added to the tour. Because of the location of i's corresponding destination, j, the tour must at sometime retrace itself in order to visit stop j, as shown in panel (b). Stop j would not sequenced until it was absolutely necessary to do so. Panel (c) presents another possible partial tour which is clearly much cheaper.

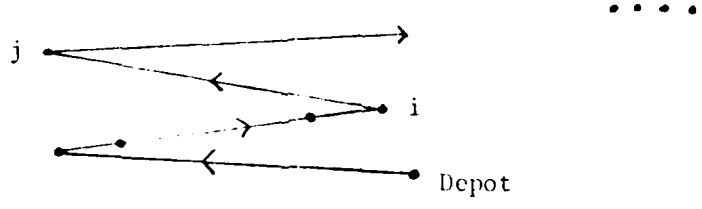
TABLE 16

HEURISTIC VERSES OPTIMAL SOLUTION VALUES ON THREE
RANDOM PROBLEM INSTANCES, N=91, Q=7, M=4

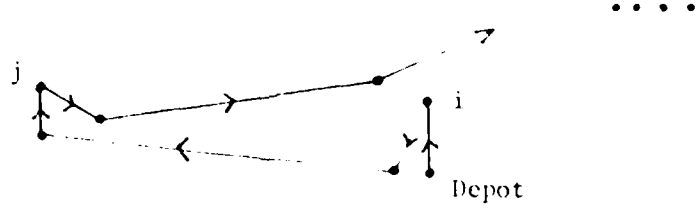
PROBLEM NUMBER	OPTIMAL	CLARKE-WRIGHT	ROUTE INSERTION	NEAREST NEIGHBOR	CLARKE-WRIGHT PLUS 3-OPTIMAL	ROUTE INSERTION PLUS 3-OPTIMAL	NEAREST NEIGHBOR PLUS 3-OPTIMAL	3-OPTIMAL ON ORDERING P	BEST OF 5 3-OPTIMAL RANDOM TOUR
1	3,842	4,972	4,840	4,294	4,112	4,134	3,946	4,086	3,996
2	3,930	5,182	4,736	4,504	4,146	4,082	4,068	4,008	3,948
3	3,694	4,942	4,426	4,056	3,718	3,990	3,786	3,858	3,756
TOTAL	11,466	15,096	14,002	12,854	11,976	12,206	11,800	11,952	11,700
HEURISTIC TOTAL / OPTIMAL TOTAL		1.316	1.221	1.121	1.044	1.064	1.029	1.042	1.020



(a) One partial tour



(b) Adding j to (a)'s tour

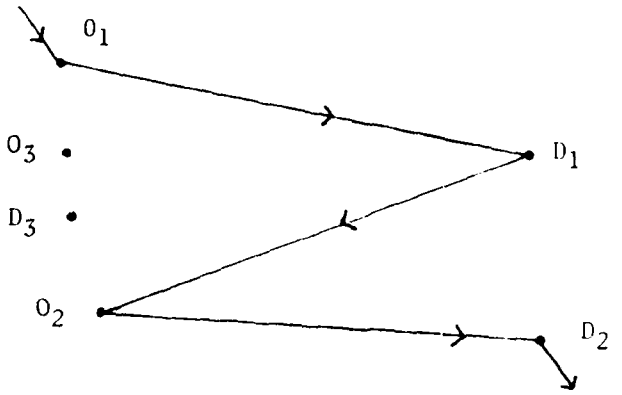


(c) Another partial tour

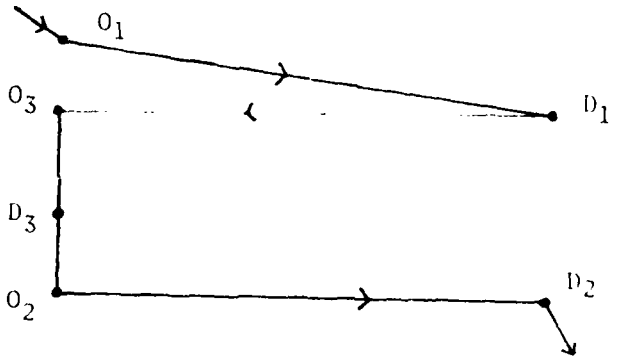
Figure 13. Example of how the precedence requirement can result in higher than necessary tour costs.

For the pair insertion heuristic, Figure 14 presents an example of the possible problems. Panel (a) depicts a partial tour. Pair number 3 is the next pair to be added. Panel (b) shows the insertion of O_3 and D_3 into the partial tour in the prescribed manner. Panel (c) shows another partial tour, which is obviously superior to that depicted in panel (b). Because the relative order of D_1 preceding O_2 in panel (a), the configuration in panel (c) could never be obtained, since the relative order there has O_2 preceding D_1.

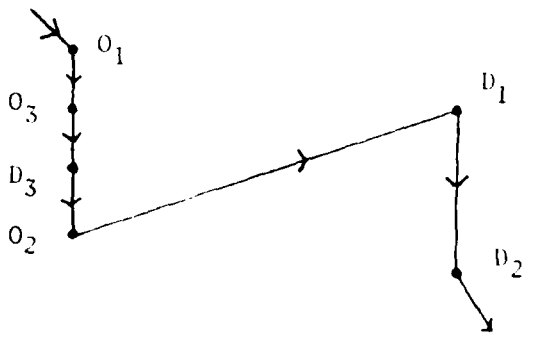
Because of the problems created by the precedence requirement in attempting to construct a tour point by point or pair by pair, it is considered doubtful that any tour construction heuristic could be developed that is superior to the 3-optimal heuristic. Further, the 3-optimal heuristic is more efficient in terms of computational effort. Therefore, any further attempt to develop tour construction heuristics for the PUDP is not recommended. In Chapter VII subject areas that appear to be much more promising for further research are suggested.



(a) Tour with pairs 1 and 2 sequenced



(b) Pair 3 added to panel (a)



(c) A superior tour for pairs 1, 2 and 3

Figure 14. An example demonstrating pair insertion problems.

CHAPTER VI

MULTIPLE VEHICLE PROBLEM

In this chapter, the algorithms used in the previous chapters to solve the single vehicle pickup and delivery problem are extended to the multiple vehicle case. The multiple vehicle problem is more difficult to solve than is the single vehicle problem. This is true both for exact and heuristic solutions to the problem. Exact solution to the problem, using the dynamic programming algorithm detailed in Chapter IV, is first discussed. Then some of the heuristic solutions detailed in Chapter V are examined.

I. EXACT SOLUTION TO THE MULTIPLE VEHICLE PROBLEM

The discussion contained in Chapter IV suggested that dynamic programming was the only exact solution algorithm that appeared promising for optimal solution of the single vehicle PUDP. The inherent complexity of the problem was the primary reason that other techniques were determined to be ineffective. Because the multiple vehicle PUDP is even more complex than the single vehicle version, these same arguments appear to apply even more strongly. Therefore, only dynamic programming was considered for extension to the multiple vehicle case.

Dynamic Programming Extended to the Multiple Vehicle Problem

Very large single vehicle PUDP's were solved using the dynamic programming algorithm developed in Chapter IV. The key factor that allowed for such success was that all problem constraints were assumed to be expressable in terms of stop numbers. The state space could, therefore, be drastically reduced by only generating feasible combinations. A binary representation vector was used to identify the specific customer points that comprised each of these vectors. The state space cannot be as drastically reduced in the multiple vehicle problem. In fact, given a reasonable quality of service parameter, Q, the constrained problems that can be optimally solved are nearly identical to their unconstrained couterparts. The reason for this lies in the manner in which the state vectors are generated and then used to determined the optimal solution.

Solution on an expanded network. In Chapter II, the solution of the multiple travelling salesman problem on an expanded network was discussed. The same technique can be used for the PUDP. The only special precaution necessary for using the dynamic programming algorithm is that one of the points representing the depot can only be considered for the lead position if the set S contains nothing other than origin/destination pairs and possibly points representing the common depot. This insures that a customer is not picked up by one vehicle and delivered by another.

Instances of interest. Consider the situation where each vehicle is to provide service to an equal number of customers. Therefore, the number of customers is assumed to be an integer multiple of the number of vehicles. For example, if there were five vehicles, then there would be either 5, 10, 15, 20, 25. . .customers. Given this scenario, those values of k where the lead position in the dynamic programming state vector is to be a depot point are well defined. Suppose there were 25 customers to be assigned among the five vehicles. Recalling that k represents the total number of points considered for inclusion, the lead positions would be the 11th, 22nd, 33rd, etc. There are ten positions to accomodate five pairs prior to the first depot point (one route); 21 positions to accomodate two sets of five pairs and the first depot point prior to the second depot point (two routes); etc.

Although this problem could be solved on the expanded network, as suggested above, a simpler method is available for this instance that requires considerably less storage than does solution on an expanded network. Unfortunately, even the simpler method requires too much storage for the solution of problems with more than about 10 origin/destination pairs.

Storage requirements for expanded network solution. Suppose only the precedence constraints are binding. Let b represent the number of origin/destination pairs assigned to a vehicle and recall that n is the number of origin/destination pairs. In the example above with five vehicles, n = 25 and b = 5. These data will be used at each step to illustrate the computations.

First, consider the number of storage locations for k values directly preceding k = 2b + 1 or 11, which is the point at which one of the depot points is first considered for the lead position. For k = 2b or 10 there are

$$n \begin{pmatrix} n-1 \\ b-1 \end{pmatrix} \tag{6.1}$$

storage locations required for the f_{10} values. The lead position must always be one of the n origins. Expression (6.1) is based on n possible origins visited immediately upon departure from the depot and the n-1 taken b-1 ways of selecting the remaining customers to be serviced by one of the vehicles. Thus, the example data result in

$$25 \begin{pmatrix} 24 \\ 4 \end{pmatrix} = 265,650$$

required locations. For k=2b-1 or 9 and an origin in the lead position, the storage required is

$$n \begin{pmatrix} n-1 \\ b-2 \end{pmatrix} \begin{pmatrix} n-b+1 \\ 1 \end{pmatrix} . \tag{6.2}$$

In terms of the example, this represents 25 choices of the lead origin, 24 remaining pairs taken 3 at a time, and one choice among the 21 remaining destinations. This destination is not a match to one of the origins being considered, but would match the origin selected subsequently at the k = 10 iteration. The value of this computation is

1,062,600. If a destination were the lead, the required storage would be the same as that given by expression (6.1). In the example, the required storage would be 1,062,600 + 265,650 = 1,328,250 for k = 9. Considering k = 16 in this same example, the storage required for just the f_{16} values would be in excess of 68 million storage locations. Clearly, practical solution to the multiple vehicle PUDP on an expanded network is limited to rather small problems.

Alternative Solution Using
Dynamic Programming

In the above discussion, once all of the f_{2b+1} or f_{11} values are computed, the data necessary to solve the problem is at hand. This is so since the f_{2b+1} values represent the cost of all feasible tours of the n customers taken b at a time. In the example, the f_{11} values contain data on all combinations of customers taken 5 at a time. Given 5 vehicles being used, the optimal solution is found by taking the 5 f_{11} values whose sum is minimal and whose combined individual state vectors indicate that service is provided to all customers. In general, this can be accomplished by a direct comparison of all of the f_{2b+1} values and their binary representation vectors. The FORTRAN routine to accomplish this requires V nested DO loops, where V represents the number of vehicles. Each loop corresponds to one of the vehicles, and an assignment of customers to that vehicle comes from the identification vector. Suppose V = 2 and the customers set is {1,2,3,4,5,6}. Therefore, three customers must be assigned to each vehicle. If the identification vector for the first loop associated with a specific

f_2 value identifies the set {1,3,4}, while the one for the second identifies {2,5,6}, their combination results in service to all of the customers. Furthermore, if the sum of the two f_7 values is minimal with respect to all other feasible combinations, the two optimal routes can be solved for individually, each as a single vehicle problem, thus solving the two vehicle problem.

The range of the exterior loop is over the number of f_{2b+1} values. For the earlier example above, there would be five loops, each with an initial range of approximately 265,650. Let F = {f_{2b+1} values}. At first look, it might appear that this routine requires on the order of $|F|^V$ comparisons or $265{,}650^5 \doteq 1.323 \times 10^{27}$ comparisons. However, the inner loops need not be entered unless there is a possibility that the optimal solution is contained within.

Requirement for no overlapping customers. The representation vectors identify the customers serviced by each one of the |F| individual tours. If the same origin/destination pair is known to be in any two of the loops, then there is no way that a feasible solution can exist. The reason for this is that each loop, in effect, assigns exactly b(N/V) customers to a vehicle. If the same customer is assigned to more than one vehicle, it must be true that at least one customer is not assigned to any vehicle. Let

R_i = representation vector of the ith individual tour. (6.3)

Suppose V > 3, then if

$$R_i \cap R_j \neq \emptyset \tag{6.4}$$

for the first two loops, the second loop can be immediately incremented since at least one origin/destination pair is contained in both partial tours. In terms of the previous example where n=25, V=5 and b=5, if one customer pair showed up in the intersection, the first two loops provide assignments for only 9 customers. The best that the three remaining loops could do is provide assignments for 15 more. Consequently, one customer is not assigned to any vehicle, and the solution is thereby infeasible. If the intersection is null, a feasible complete tour can be found and the third loop is entered. Let R_k be a vector for this third loop. Now, if

$$(R_i \cap R_k) \cup (R_j \cap R_k) \neq \emptyset \tag{6.5}$$

the third loop is incremented and another R_k selected. Otherwise the fourth loop is entered. Finally, the Vth loop is entered, a representation vector is found such that all customers are serviced. This represents a feasible solution and the cost of this solution, represented by the V f_{2b+1} values, is computed. If the value of this solution is better than the incumbant, it becomes the new solution. The innermost loop is then incremented until another solution is found or until the loop parameter is exhausted. These basic steps are continued until the lowest cost feasible complete tour is found.

FORTRAN intersection. Graves (20) provides techniques for performing logical operations in FORTRAN without recourse to assembly level routines. His approach is an order of magnitude quicker than using an assembly level routine since it eliminates several storage and retrieval operations for register values as the program shifts control from the main routine to the subroutine and back again. The EQUIVALENCE statement is the key as shown in the following algorithm which computes the binary intersection A of two representation vectors B and C and prints out the decimal equivalent:

```
INTEGER A,B,C
LOGICAL*4 LA,LB,LC
EQUIVALENCE (A,LA),(B,LB),(C,LC)
LA = LB.AND.LC
PRINT A
```

Thus, if A = 0, the intersection is null. If $A \neq 0$, two or more origin/destination pairs are continued in both B and C. Identification of which pairs are redundant could be found by converting the decimal value of A into its binary equivalent.

Limitation on Problem Size

As mentioned above, exact solution to problems with more than about ten origin/destination pairs is not practical in the case of the multiple vehicle PUDP. Time windows can still be used to determine those points which are valid condidates for the lead position or which are eligible to be used to complete a given state vector. They cannot be used to require points to be included in the completion of a state vector. A point which, if not sequenced would preclude any feasible

solution in the single vehicle problem, can be assigned to any of the V vehicles in the multiple vehicle case. For the alternative solutions approach discussed above, there is never an instance where a point must be included. The feasible state space is also larger due to the fact that the candidate set for the lead position is approximately V times greater than it is for the single vehicle problem. Consequently, the number of feasible states for each value of k and choice of lead is considerably greater.

Computational Results

Table 17 presents typical results for various sized problems. Only the precedence constraints were binding. The largest of these problems actually solved involved only nine origin/destination pairs and three vehicles. This equates to a 21 point multiple travelling salesman problem on an expanded graph. Attempting to add one more customer per vehicle resulted in exceeding a five minute execution time limit. In fact, the limit was exceeded during the computation of one of the 9,900 f_5 values. This means the problem was less than one half solved at the time of termination, since storage required up to and including f_5 is only 14,994 locations.

Constrained problems produced very similar results. The 12 customer, 3 vehicle problem was too large to solve. Using a value of Q=5, the time windows for the remaining problems are very similar to those for an unconstrained problem. Consequently, the number of feasible states generated is not significantly reduced. Table 18 provides a side

TABLE 17

SELECTED OPTIMAL RESULTS FOR THE MULTIPLE VEHICLE PICKUP AND DELIVERY PROBLEM

NUMBER OF VEHICLES	NUMBER OF CUSTOMERS	N	STORAGE REQUIRED	RUN TIME (SEC)
2	4	9	65	3.1
2	6	13	667	4.3
2	8	17	6,337	74.53
3	6	13	157	3.2
3	9	19	2,566	22.4
3	12	25	39,414	> 300

TABLE 18

COMPARISON OF TIME WINDOWS FOR CONSTRAINED VERSUS UNCONSTRAINED VERSION OF THREE VEHICLE, NINE CUSTOMER FUDP

(a) UNCONSTRAINED EQUIVALENT						(b) CONSTRAINED Q=5					
ORIGINS			DESTINATIONS			ORIGINS		DESTINATIONS			
#	NET	NLT	#	NET	NLT	NET	NLT	#	NET	NLT	
2	1	5	11	2	6	1	3	11	2	4	
3	1	5	12	2	6	1	3	12	2	4	
4	1	5	13	2	6	1	4	13	2	5	
5	1	5	14	2	6	1	4	14	2	5	
6	1	5	15	2	6	1	4	15	2	5	
7	1	5	16	2	6	1	5	16	2	6	
8	1	5	17	2	6	1	5	17	2	6	
9	1	5	18	2	6	1	5	18	2	6	
10	1	5	19	2	6	1	5	19	2	6	

by side comparison for the 9 customer, 3 vehicle problem. While the unconstrained problem required 2,566 storage locations, the constrained counterpart requires 2,128 locations. Consequently, optimal solution to any real world multiple vehicle problem is not deemed practical. The state space becomes too large, just as it does in the travelling salesman problem.

II. HEURISTIC SOLUTION TO THE MULTIPLE VEHICLE PROBLEM

The multiple vehicle PUDP was shown in the preceding section to be more difficult to solve optimally. It is also more difficult to obtain a good solution by means of a heuristic, although a precise measure of "goodness" is difficult because only small problems can be solved optimally. For the single vehicle problem, the 3-optimal heuristic produced solutions generally within a few percentage points of the optimal solution. As will be seen below, the 3-optimal technique is severely limited because of the precedence relationship in the multiple vehicle problem.

The key to good solutions in the multiple vehicle problem is determining which vehicle customers should be assigned to. Once customers are assigned to a given vehicle, the problem reduces to V single vehicle problems for which optimal or very good solutions are readily attainable. Several attempts were made to determine a good method to make such assignments. The results are somewhat discouraging as will be seen later. Consequently, the multiple vehicle problem appears to remain relatively unsolved.

Clarke-Wright and Pair Insertion

Two of the tour construction heuristics discussed in the last chapter were the Clarke-Wright savings heuristic and the pair insertion heuristic. Both of these heuristics were expensive in terms of computational effort and performed relatively poorly. Other heuristics produced much better results. Therefore, neither of these approaches was pursued as a possible solution to the multiple vehicle problem.

Greedy Heuristic

The greedy heuristic is also a tour construction heuristic, but is much less expensive computationally. The heuristic also has the advantage that it is easily understood and can often be implemented manually. Therefore, the greedy heuristic was selected to determine how well a tour construction heuristic performed on the multiple vehicle problem. The results were surprisingly good.

Construction concept. There are two ways that tours could be constructed using the nearest neighbor concept. One would be a sequential approach where vehicles are dispatched one at a time. In this case, 1/V of the customers would be assigned to the first vehicle before the second is dispatched. The second approach dispatches all vehicles at the same time and builds routes simultaneously. The latter approach was used for the multiple vehicle PUDP, since it produced individual routes of nearly equal length. The former method tends to produce at least one relatively short route and at least one relatively long route which may be undesirable.

Description of algorithm. Initially, the V nearest origins to the depot which satisfy the time window constraints are assigned one each to the V vehicles. Then a second stop is determined for all the vehicles, followed by the third, fourth, etc. A candidate set is used to identify all unsequenced points that satisfy the time window constraints. For each vehicle, the next nearest feasible neighbor from this candidate set is added to the partial tour, provided a check, made to insure that adding the nearest neighbor will result in a feasible tour, is successful.

Feasibility check. Insuring that points added allow for a feasible set of individual tours is essential for the multiple vehicle problem, just as it was for the single vehicle problem. The construction or flushing out of the partially completed tours is a bit more complex in the multiple vehicle problem. The nondecreasing ordering P is used, just as in the single vehicle case. The next element of P, not already sequenced by the greedy selection, is placed on the vehicle with the fewest elements unless it is a destination. Destinations must be placed on the same vehicle as their corresponding origins. Each of the V individual tours is then checked to verify that it is feasible. Unless all V of the tours are feasible, the nearest neighbor being considered for addition to one of the tours is not added. Rather it is removed from the candidate set for that vehicle and another candidate (the nearest remaining feasible neighbor) is selected and the feasibility check repeated.

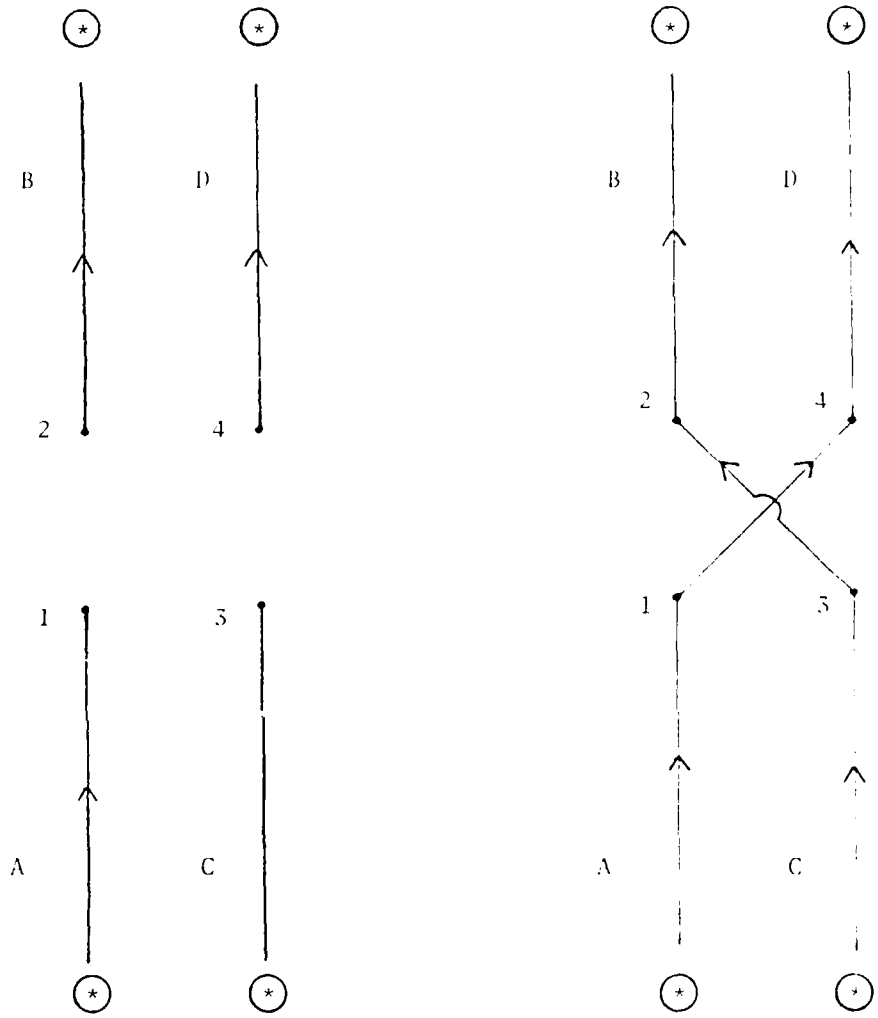
Optimal individual tours. Once all V tours have been constructed, it is often possible to determine the optimal routing for each tour. This is possible when all problem constraints are expressable in terms of the vehicles' stop numbers as discussed in Chapter IV. The greedy solution identifies which pairs are on each vehicle, thus defining V single vehicle problems. If the constraints are expressed in a manner such that the dynamic programming algorithm can not be used, or if the problem is too large for the dynamic programming algorithm, the 3-optimal heuristic can be applied to the greedy solution. These techniques were applied to all of the problems studied herein.

Greedy look ahead. The basic greedy algorithm is myopic in that one selects the nearest neighbor without thought of where the tour will next go. The precedence requirement of the PUDP necessitates subsequent travel to the as yet unsequenced destinations. It was conjectured that by considering (minimizing) the distance over the next two points that a better solution could be obtained, especially in terms of which pairs were assigned to which vehicle. The basic greedy algorithm is an order Z algorithm, where Z represents the cardinality of the current candidate set. The look ahead greedy is approximately an order Z^2 algorithm. Results indicated no appreciable difference between the greedy and the look ahead greedy. Therefore, the look ahead version was not tested further. The heavier computational effort could not be justified.

r-Optimal Heuristic

Limitations. The results of Chapter V clearly showed the superiority of the 3-optimal heuristic when compared to any of the tour construction heuristics. However, the precedence constraint nullifies much of the effectiveness of the r-optimal heuristic when applied to the multiple vehicle problem. The solution presented to the 3-optimal algorithm is on an expanded network. Since a very good (if not optimal) solution can be obtained for the individual tours, interchanging arcs between tours and thus creating a change in the customers assigned to a given vehicle is what is desired. The problem stems from the necessity that both the origin and destination be on the same vehicle. For the pickup and delivery problem, this requirement can only be met when the vehicles are empty, as is shown below.

Reconnection pattern. The crux of the problem can be readily seen by considering a 2-optimal example. In Figure 15(a), the two individual tours, each with an arc removed, are shown side by side. The four (*)'s all represent the common depot, and the individual tours begin at the bottom and proceed upward. Each letter represents the set of points between the depot and a removed arc. Figure 15(b) depicts the only available reconnection pattern. Let i and j represent an origin and its corresponding destination respectively. For the configuration in Figure 15(a) to be initially feasible the following conditions must hold:



(a) Original tours with two arcs removed

(b) New tours after reconnection

Figure 15. Pictoral of 2-optimal reconnection pattern for the multiple vehicle problem.

$$i \in A \Rightarrow j \in (A \cup B) \quad , \tag{6.6}$$

$$i \in C \Rightarrow j \in (C \cup D) \quad , \tag{6.7}$$

$$j \in B \Rightarrow i \in (A \cup B) \quad , \tag{6.8}$$

and

$$j \in D \Rightarrow i \in (C \cup D) \quad . \tag{6.9}$$

Clearly, if

$i \in A$ and $j \in B$ or if $i \in C$ and $j \in D$

removal of either of these arcs could never result in a feasible solution, as reconnection would result in the origin assigned to one vehicle and the destination to another. Only when the vehicle is empty can an arc feasibly be removed. In order to maintain an equal number of customers serviced by each vehicle, both vehicles must become empty at the same stop number. The likelihood of this occurring is rather small based on the empirical evidence. Even when such an occurance does materialized, the interchange is seldom profitable.

Extension to 3-optimal. Given the restriction that each vehicle must provide service to the same number of customers, there is only one way three arcs can be feasibly removed. Removal of two arcs from one tour alters the number of customers served by a given vehicle. Therefore, the arcs must come from each of three separate tours, all at the same vehicle stop number, and with all vehicles empty prior to the break.

The likelihood of this occurring is even less than with a two arc removal. Preliminary results suggested the approach was not effective in solving the multiple vehicle PUDP and was not pursued further. This is but another example of how the precedence relationship of the PUDP severely limits the effectiveness of heuristics that otherwise work well on related problems.

Pair Selection

The pair selection heuristic identifies the profitability of assigning each pair of origin/destination pairs to the same vehicle. Using a savings value computed for each pair of pairs, a group of customers is eventually identified for each of the V vehicles. Once the V groups are identified, the problem is reduced to V single vehicle problems, each solved either optimally or by the 3-optimal heuristic.

Concept. The motivation for the pair selection heuristic lies in taking advantage of the requirement to move from an origin to its corresponding destination. If two customers can be served as cheaply as one, or nearly so, it appears advantageous to do so. Figure 16 shows two origin/destination pairs that can both be serviced for the same cost as serving the first alone. Therefore, the savings value is computed to determine how much can be saved by serving two customers on the same route. This value is termed SAV.

SAV formulas. For any pair of customers taken by themselves there are only six ways in which they could be serviced. Suppose both

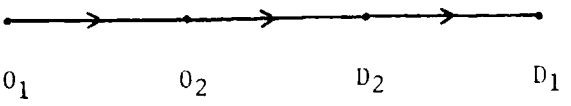


Figure 16. Example of how two customers can be served at the cost of one.

customers are considered separately, and the only cost for each is that from its origin directly to its destination. How much could be saved by having both customers on the same route? The savings is taken to be the cost of linking the two origin/destination pairs minus the cost of separate service. The SAV value, which may be either positive or negative, is taken to be the minimum cost of each of the six possible service patterns. Table 19 gives the six different reconnection or service patterns and the savings formulas in FORTRAN symbology for each. The blank spaces result from costs that appear with both a positive and a negative sign in the same formula and are thus cancelled out.

Selection procedure. The SAV value that is the most negative (least positive) identifies the first two origin/destination pairs to be assigned to a vehicle. The next smallest SAV value identifies the next two pairs. Provided neither is assigned to the first vehicle, they are assigned to the second vehicle. If one had been assigned to the first vehicle, the other would be also. This would leave the first vehicle with three assigned customers. With only precedence constraints binding, this process is continued until at least one customer (origin/destination pair) is assigned to each of the V vehicles.

Subsequent pair assignment. Once all V vehicles have been assigned at least one customer, the process becomes slightly more complex. Let I and J represent the next pair to be considered based on the SAV(I,J) being the minimum of the remaining values. If I has already been assigned but J has not, J is assigned to the same vehicle as I is,

TABLE 19

PAIR SELECTION PATTERNS AND
SAVINGS FORMULAS

	PATTERN	SAVING FORMULA
(1)	I-J-ID-JD	C(I,J)+C(J,ID)+C(ID,JD)-C(I,ID)-C(J,JD)
(2)	I-J-JD-ID	C(I,J) +C(JD,ID)-C(I,ID)
(3)	I-ID-J-JD	C(ID,J)
(4)	J-I-JD-ID	C(J,I)+C(I,JD)+C(JD,ID)-C(I,ID)-C(J,JD)
(5)	J-I-ID-JD	C(J,I) +C(ID,JD) -C(J,JD)
(6)	J-JD-I-ID	C(JD,I)

NOTE: I, the first origin
J, the second origin
C(·,*), Cost or distance from · to *
ID, I's destination
JD, J's destination

provided the vehicle does not already have its full complement of customers. If I has been assigned to one vehicle and J to another, the pair is already taken care of by previous assignments. If neither can be matched with a vehicle at this time, the pair is held until one of the customers is assigned to one of the V vehicles. Then the other is assigned to the same vehicle provided that the first customer did not complete the vehicle's complement. Also, when time windows are present, it is necessary to insure that each assignment can lead to feasible individual tours.

Feasibility of assignments. Insuring feasible tours given the presence of time windows has been a critical factor in all of the PUDP heuristics examined. The pair selection procedure is no exception. The procedure parallels that used in both the single vehicle cases and the multiple vehicle case by greedy selection. V individual tours are constructed using the ordering P and then each tested for feasibility. The customer or customers being considered for service by a given vehicle are assigned to that vehicle only if all V of the tours are feasible.

Interchange Heuristic

Concept. Given V feasible individual tours, the interchange heuristic attempts to identify two customers on different tours that can be switched so as to reduce the combined cost of the two respective tours. This is accomplished by first determining for each customer the

penalty associated with being on the tour it is on and not on one of the other tours, and then matching the penalties to find those that appear to be good candidates for a switch.

Penalty computation. The penalty for origin I and its corresponding destination ID on tour k not being on tour k' is defined as the difference between the cost of connecting to this pair in its present tour and the minimum cost to connect this pair to tour k'. Let IP(IDP) and IF(IDF) represent the stops directly preceding and directly following I(ID) in its present tour. Also let J and J' with J preceding J' represent the closest points to I and ID on tour k' respectively. Then the penalty can be computed by

$$\min [C(IP,I)+C(I,IF)+C(IDP,ID)+C(ID,IDF);$$
$$C(IP,I)+C(I,ID)+C(ID,IDP)]$$
$$-2[C(I,J)+C(ID,J')] \qquad (6.10)$$

Large penalty values suggest it may be better to have the pair on the other tour.

Switching pairs. It would not be practical to attempt all of the possible switches of two pair. Therefore, switches of the pairs comprising the highest four or five penalty values were attempted. The switch effects a change in the composition of each vehicle's customers. Therefore, the new configurations are optimally solved for both individual tours. If the combined tour cost is less after the switch, the switch becomes permanent and the process repeated.

Interchange results. The heuristic was tried on several problems of varying size and complexity. The heuristic very seldom identified a profitable switch when the initial tours were the optimal routing obtained from either the greedy or the pair selection procedures. Consequently, this heuristic was eliminated from further consideration.

Computational Results

Because it is much more difficult to obtain optimal solutions for the multiple vehicle PUDP than it is for the single vehicle version, only in the case of very small problems can the heuristics be compared to the optimal solution. For larger problems, results are limited to a relative comparison. As in the single vehicle case, the impact of varying the service paramenters is also investigated.

Heuristics verses optimal. Table 20 presents results from five sample problems. These data indicate that the pair selection technique is very effective for problems of this size. The largest deviation from optimal was less than 5%. The data also suggests that the greedy solution followed by optimally sequencing the individual routes is not especially effective. Deviations of greater than 20% are noted. However, as will be seen later, this greedy procedure is still superior to a random assignment of the customers among the vehicles. Also noted below is the fact that on larger problems with time windows present, the greedy selection heuristic demonstrates equal or slightly better performance when compared to the pair selection procedure.

TABLE 20

COMPARISON OF MULTIPLE VEHICLE HEURISTIC
TO THE OPTIMAL SOLUTION VALUES

N	V	OPTIMAL	GREEDY	GREEDY SELECTION	PAIR SELECTION
17	2	604	918	734	604
17	2	688	742	730	704
17	2	530	746	598	536
13	3	802	1096	974	802
13	3	794	926	808	808

Comparison of greedy and pair selection on unconstrained problems. When there are no time window constraints present, the pair selection procedure generally produces better results than does the greedy selection procedure. The data in Table 21 verifies this for ten problems with 16 customers (N=33) and four vehicles. Only in one instance (problem 8) was the greedy pair solution better than the pair selection solution. Only a relative comparison is possible since N=33 and V=4 is too large to be optimally solved by dynamic programming. Consequently, these results may or may not be anywhere near optimal. The questions of whether these selection procedures are superior to a haphazard selection procedure remains to be answered.

Selection comparison on unconstrained problems. Table 22 presents ten solutions to each of two problems with only precedence constraints binding. The first and second solutions for each problem are the solutions obtained from the greedy selection and pair selection heuristics respectively. Solutions three through ten represent a random assignment of four customers to each of the four vehicles. In all cases, each vehicle is individually optimally routed using the dynamic programming algorithm. Unquestionably, both the greedy and pair selection procedures produce better results than just randomly or haphazardly assigning customers to the vehicles. It is also evident that the key to a good solution for the multiple vehicle PDPP is in the assignment of customers to the vehicles. This is especially true for those instances that allow for an optimal sequencing of the assigned pairs. Such an assignment is complicated since there are

TABLE 21

COMPARISON OF GREEDY AND PAIR SELECTION TECHNIQUES FOR THE UNCONSTRAINED PUDP, N=33, V=4

PROBLEM NUMBER	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	1564	1362
2	1486	1444
3	1280	1272
4	1488	1352
5	1386	1250
6	1500	1390
7	1570	1328
8	1194	1306
9	1492	1452
10	1534	1402

TABLE 22

COMPARISON OF SOLUTION VALUES BASED ON
DIFFERENT SELECTION OF CUSTOMERS,
N=33, V=4

SOLUTION NUMBER	PROBLEM 1 SOLUTION	PROBLEM 2 SOLUTION
1 (Greedy)	1564	1488
2 (Pair)	1362	1444
3	1804	1609
4	1792	1596
5	1736	1492
6	1758	1470
7	1730	1596
8	1692	1534
9	1818	1634
10	1766	1538

$$\begin{pmatrix} n \\ n / v \end{pmatrix} \tag{6.11}$$

ways to distribute the customers among the V vehicles. For the problems discussed immediately above this represents

$$\begin{pmatrix} 16 \\ 4 \end{pmatrix} = 1,820 \tag{6.12}$$

different possibilities.

Varying service parameters. The previous discussion assumed that only the precedence constraints were binding. When time windows are present as they are in the dial-a-ride service problem, the apparent superiority of the pair selection heuristic disappears. Tables 23 through 26 present results for 16 customers (N=33), four vehicles and varying values of the quality of service parameters Q and M. The solutions are presented for three solution methods: greedy, greedy pair selection, and pair selection. Clearly, the individual greedy tours should be subjected to the optimal or r-optimal algorithms to improve the initial tour. When the quality of service constraints are tight, as shown in Tables 23 and 24, the greedy selection procedure shows a slight superiority over the pair selection technique. As the constraints become less binding, the preference swings toward the pair selection heuristic. However, for larger problems in the constrained environment, the data do not support either heuristic being superior to the other.

TABLE 23

COMPARISON OF SOLUTION VALUES FOR THE GREEDY, GREEDY PAIR AND PAIR SELECTION HEURISTICS, N=33, V=4, Q=5, M=3

PROBLEM NUMBER	GREEDY SOLUTION	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	1962	1932 *	2004
2	2190	2084 *	2138
3	1894	1894 *	1998
4	1974	1958 *	1962
5	1900	1848 *	1988
6	1866	1808 *	2100
7	2612	2576	2566 *
8	2308	2048 *	2112
9	1984	1972	1914 *
10	2066	1930 *	2260

*Best Solution

TABLE 25

COMPARISON OF SOLUTION VALUES FOR THE GREEDY, GREEDY PAIR AND PAIR SELECTION HEURISTICS, N=33, V=4, Q=5, M=5

PROBLEM NUMBER	GREEDY SOLUTION	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	2126	1932 *	1984
2	2036	1960 *	2054
3	1864	1832 *	1942
4	2120	1894 *	1894
5	1866	1802 *	1950
6	1776	1720 *	1994
7	2198	2166 *	2354
8	2286	2144	1976 *
9	1948	1862	1800 *
10	2096	2016 *	2164

*Best Solution

TABLE 25

COMPARISON OF SOLUTION VALUES FOR THE GREEDY, GREEDY PAIR AND PAIR SELECTION HEURISTICS, N=33, V=4, Q=7, M=4

PROBLEM NUMBER	GREEDY SOLUTION	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	1782	1716	1580 *
2	1566	1548 *	1638
3	1988	1740	1518 *
4	1764	1650	1484 *
5	1668	1624	1502 *
6	1588	1538 *	1568
7	1668	1620 *	1724
8	1644	1526	1480 *
9	1880	1604	1562 *
10	1772	1654	1624 *

*Best Solution

TABLE 26

COMPARISON OF SOLUTION VALUES FOR THE GREEDY, GREEDY PAIR AND PAIR SELECTION HEURISTICS, N=33, V=4, Q=11, M=6

PROBLEM NUMBER	GREEDY SOLUTION	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	2062	1684	1618 *
2	1518	1462 *	1560
3	1602	1470	1568 *
4	1932	1584	1482 *
5	1876	1558 *	1584
6	1596	1504	1444 *
7	1770	1728	1580 *
8	1802	1522 *	1458
9	1754	1624 *	1636
10	1740	1576	1486 *

*Best Solution

Performance on larger problems. Tables 27 through 31 present results for 30 customers and either three or five vehicles. The results parallel those discussed above. Neither the pair selection nor the greedy selection produces consistently superior results. When the constraints are relatively tight, the greedy selection procedure appears to produce better results.

Discussion of results. The fact that a greedy-like procedure often produces the best results is uncomforting. Greedy heuristics seldom produce the best solutions on other difficult combinatorial problems. The fact that these solutions cannot be compared to the optimal is also distressing. There is no way of telling the true effectiveness of these heuristics on other than very small problems. The pair selection heuristic, as developed, and the greedy selection heuristics may well be excellent ones. However, it seems likely that a better technique for determining which customers to assign to what vehicle exists. What that technique is or what the basis is for its development are as yet unanswered questions. Such questions represent one area for further investigation. This and other areas that appear to offer research opportunities that expand on this work are discussed in the next chapter.

TABLE 27

COMPARISON OF SOLUTION VALUES FOR THE GREEDY, GREEDY PAIR AND PAIR SELECTION HEURISTICS, N=61, V=3, Q=5, M=5

PROBLEM NUMBER	GREEDY SOLUTION	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	3696	3640 *	3932
2	3558	3396 *	3824
3	3780	3534 *	3636
4	3234	3130 *	3238
5	3354	3134 *	3316
6	3410	3374 *	3544
7	3036	3032 *	3242
8	3505	3408 *	3428
9	3662	3576	3468 *
10	3652	3622 *	3940

*Best Solution

TABLE 28

COMPARISON OF SOLUTION VALUES FOR THE GREEDY, GREEDY PAIR AND PAIR SELECTION HEURISTICS, N=61, V=3, Q=7, M=4

PROBLEM NUMBER	GREEDY SOLUTION	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	2940	2824 *	2838
2	3230	2912 *	2994
3	2876	2730 *	2852
4	3088	2870	2628 *
5	2902	2606 *	2772
6	2872	2668 *	2814
7	2876	2646 *	2676
8	2708	2572 *	2768
9	3514	2922	2712 *
10	3214	2950	2906 *

*Best Solution

TABLE 29

COMPARISON OF SOLUTION VALUES FOR THE GREEDY, GREEDY PAIR AND PAIR SELECTION HEURISTICS, N=61, V=5, Q=11, M=6

PROBLEM NUMBER	GREEDY SOLUTION	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	3198	2852	2584 *
2	2984	2736	2708 *
3	2830	2688 *	2858
4	2962	2708 *	2850
5	2598	2240 *	2394
6	2390	2218 *	2510
7	2888	2758	2550 *
8	2710	2506	2420 *
9	2946	2738 *	2964
10	3260	2838	2554 *

*Best Solution

TABLE 30

COMPARISON OF SOLUTION VALUES FOR THE GREEDY, GREEDY PAIR AND PAIR SELECTION HEURISTICS, N=61, V=5, Q=11, M=11

PROBLEM NUMBER	GREEDY SOLUTION	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	3226	2826	2584 *
2	2888	2702	2670 *
3	2984	2758 *	2784
4	2722	2612 *	2794
5	2464	2094 *	2314
6	2728	2462 *	2508
7	2804	2698 *	2728
8	2626	2376 *	2570
9	2950	2722 *	3042
10	3198	2772	2438 *

*Best Solution

TABLE 31

COMPARISON OF SOLUTION VALUES FOR THE GREEDY, GREEDY PAIR AND PAIR SELECTION HEURISTICS, N=61, V=5, Q=7, M=4

PROBLEM NUMBER	GREEDY SOLUTION	GREEDY PAIR SOLUTION	PAIR SELECTION SOLUTION
1	2890	2564 *	2840
2	3126	2796	2752 *
3	2782	2670 *	2924
4	2964	2866 *	2950
5	3200	2920	2710 *
6	2558	2452 *	2664
7	2862	2752	2722 *
8	3084	2616 *	2690
9	3076	2932	2850 *
10	3246	2996	2764 *

*Best Solution

CHAPTER VII

SUGGESTED AREAS FOR FURTHER RESEARCH

The work documented in the preceding chapters represents one of the first attempts to define and solve the pickup and delivery problem. As such, it represents only a first step in that direction. Many aspects of the problem remain to be investigated. Therefore, this chapter outlines several research opportunities that have suggested themselves during the course of this effort.

I. SUMMARY OF ASPECTS STUDIED

In Chapters I and III the PUDP was defined and discussed in its most general form. Subsequently, a more restricted form of the problem was developed and explored. Specifically, Chapters IV through VI dealt with instances of the PUDP where the constraints were expressable in terms of the stop or sequence number of the vehicle serving a particular customer. Capacity was assumed to not be a binding constraint. This allowed for the exact solution of very large single vehicle problems as well as modest sized multiple vehicle ones. The heuristics discussed considered the same problem instances in order to have a precise measure of how well a given heuristic performed. Relaxation of these restrictions offer the first area of opportunity.

II. CONSIDERATION OF THE GENERAL PROBLEM

The most general version of the PUDP is constrained by vehicle capacity, time windows, quality of service, operational considerations and of course the precedence requirement. When these constraints cannot be expressed as discussed above, the heuristic techniques developed herein will not work without modifications. The dynamic programming algorithm will not work at all in most cases. Consequently, exact solutions to the general problem appear doubtful. Therefore, comparison among heuristics probably will have to be made on a relative basis.

III. QUESTION OF THE EXISTENCE OF FEASIBILITY

Given the more general problem, one question that must be addressed is that of the existence of a feasible route or set of routes. Assuming that an r-optimal heuristic is to be used, the existence of an initial feasible solution is critical. Finding such a solution may well be an extremely difficult task for those problem instances with relatively tight constraints. Use of any tour construction heuristic implicitely requires a guarantee that a given partial tour can be extended to a feasible, complete route. Such a determination was often complicated for the problems discussed in previous chapters. It would appear that it would be even more complicated in the more general case. Notwithstanding, this represents the logical next step in studying the PUDP.

IV. MULTIPLE VEHICLE EXTENSIONS

The multiple vehicle PUDP is much more complex and difficult to solve than is the single vehicle PUDP. It also offers the greatest potential for research. The results of Chapter VI suggest that better solutions techniques may exist for the multiple vehicle problem.

Vehicle Assignment

Given a good algorithm for solving the single vehicle PUDP, the key to a good solution to the multiple vehicle problem lies in the distribution of the customers among the vehicles. Although the techniques discussed is Chapter VI did not perform as well as hoped, the basic concept still appears fruitful. That is, obtain a good initial assignment and then improve the solution by switching customers among the vehicles. The problem is in finding more effective techniques for doing so.

Number Per Vehicle

The results for the multiple vehicle problem assumed that the same number of customers would be assigned to each vehicle. Such an assignment may not be the most efficient. Consequently, relaxing this restriction offers another area for consideration.

V. POSSIBILITY OF SLACK PERIODS

In all the cases considered, there were always enough customers so that the vehicle was always in use. No consideration was given to ...

case where the vehicle might be idle awaiting the earliest time the next customer could be picked up. Especially in the dial-a-ride service problem, such a scenario is not unrealistic and should be considered. Other possibilities for slack periods might include lunch breaks or coffee breaks for the drivers.

VI. OPERATIONAL CONSTRAINTS

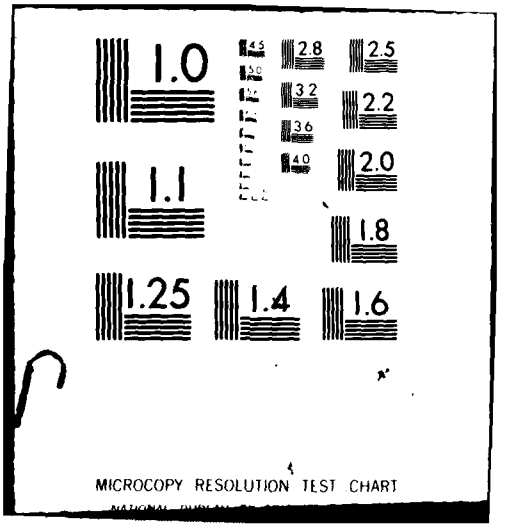
The possibility of limitations on the total time or distance that a vehicle could be operated is a possible constraint that was not addressed in Chapter VI. Such operational constraints could apply to the single vehicle problem, but with optimal or near optimal solutions possible, imposing them could result in no feasible solution. The more likely scenario for operational constraints would be in the multiple vehicle problem. If and when more effective assignment and improvement procedures are developed for the multiple vehicle problem, consideration of operational constraints would represent a possible subsequent extension.

VII. POSSIBLE HEURISTICS FOR THE SINGLE VEHICLE PDDP

Any number of additional heuristic procedures could be developed and applied to the single vehicle PDDP. However, two heuristics appear to offer good potential for all instances of the PDDP. One is an extension of the 3-optimal heuristic, while the other uses the optimal solution to a related problem.

AD-A107 202 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH F/G 12/1
THE SINGLE AND MULTIPLE VEHICLE PICKUP AND DELIVERY PROBLEM: EX--ETC(U)
JUN 81 G R ARMSTRONG
UNCLASSIFIED AFIT-CI-81-50D NL

END
DATE FILMED
12-81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

Four and 5-Optimal

For the TSP and the VRP, an r value of 3 in the r-optimal heuristic was usually the largest value used. The reason is becuase of the exponential increase in computations as the value of r increases. Consequently, values of r greater than 3 were deemed practical. The number of reconnection patterns also increases exponentially as the value of r increases. The same phenomenon is true with the PUDP. However, due to the precedence constraints of the problem, it may not be impractical to use larger values of r. This might be accomplished by only considering those reconnection patterns that offer the highest probability of obtaining a feasible reconnection. The range of stop numbers included in each loop might also be limited so as to include only those stops which are most likely to yield a feasible reconnection pattern. For the 3-optimal solution, this range was taken to be Q stops. Given the demonstrated superiority of the 3-optimal solution over the other heuristics tested, a 4 or 5-optimal heuristic appears to be a most fruitful area to investigate.

Optimal Solution to Related Problem

For the general problem where dynamic programming cannot be used to obtain the exact solution, it may be possible to use dynamic programming to obtain a good solution. This could be accomplished by approximating the general constraints by stop numbers and then solving the related problem by the dynamic programming algorithm. If the resulting solution is feasible to the original problem, one would

hopefully have a good solution. If the solution were not feasible, some interchange routine would be needed to find a feasible solution with a minimum of additional cost. Intuitively, this procedure should perform well and could be used on either the single vehicle problem or as the routing portion of a two step solution of the multiple vehicle PUDP.

VIII. PUDP POTENTIAL

The PUDP is a new problem which is only just beginning to attract research attention. Given both the complexity of the problem, which makes obtaining solutions difficult, and the practicality of the applications, the PUDP should appeal both to the theoretician and to the practitioner for some time to come. The work documented herein as well as the few related articles represent only the tip of the iceberg. A great deal remains to be accomplished before the PUDP can be considered fully solved.

LIST OF REFERENCES

LIST OF REFERENCES

1. Balinski, M. and R. Quandt, "On an Integer Program for a Delivery Problem," Operations Research, Vol. 12, No. 2, 1964, pp. 300-304.

2. Bellmore, M. and J. Malone, "Pathology of Traveling-Salesman Subtour Elimination Algorithms," Operations Research, Vol. 19, 1971, pp. 278-307.

3. Bellmore, M. and G. Nenhauser, "The Traveling Salesman Problem: A Survey," Operations Research, Vol. 16, No. 3, 1968, pp. 538-558.

4. Christofides, N., A. Mingozzi and P. Toth, "Exact Algorithms for the Vehicle Routing Problem," Unpublished Paper, Undated.

5. Christofides, N. and S. Eilon, "Algorithms for Large-Scale Travelling Salesman Problems," Operational Research Quarterly, Vol. 23, No. 4, 1972, pp. 511-518.

6. Christofides, N. and S. Eilon, "An Algorithm for the Vehicle Dispatching Problem," Operational Research Quarterly, Vol. 20, No. 3, 1969, pp. 309-318.

7. Clarke, G. and J. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operations Research, Vol. 12, No. 4, 1964, pp. 568-581.

8. Dantzig, G. and J. Ranser, "The Truck Dispatching Problem," Management Science, Vol. 3, No. 4, 1957, pp. 407-430.

9. Driscoll, W. and H. Emmons, "Vehicle Routing for Minimum Cost," Unpublished Paper, May 1979.

10. Fisher, M. and R. Jaikumar, "A Decomposition Algorithm for Large-Scale Vehicle Routing," Unpublished Paper, July 1978.

11. Fox, K., B. Gavish and S. Graves, "An n-Constraint Formulation of the (Time Dependent) Traveling Salesman Problem," Unpublished Paper, March 1979.

12. Garey, Michael R. and David S. Johnson, Computers and Interactability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, San Francisco, 1979.

13. Garfinkel, R. and G. Nemhauser, Integer Programming, John Wiley, New York, 1972.

14. Gaskell, T., "Bases for Vehicle Fleet Scheduling," Operational Research Quarterly, Vol. 18, No. 3, 1967, pp. 281-295.

15. Gillett, B. and L. Miller, "A Heuristic Algorithm for the Vehicle-Dispatch Problem," Operations Research, Vol. 22, No. 2, 1974, pp. 340-349.

16. Golden, B., "Evaluating a Sequential Vehicle Routing Algorithm," AIIE Transactions, Vol. 9, No. 2, 1977, pp. 204-208.

17. Golden, B., L. Bodin, T. Doyle, and W. Stewart, Jr., "Approximate Traveling Salesman Algorithms," Unpublished Paper, Nov 1978.

18. Golden, B., T. Magnanti and H. Nguyen, "Implementing Vehicle Routing Algorithms," Networks, Vol. 7, No. 2, 1977, pp. 113-148.

19. Gomory, R., "Outline of an Algorithm for Integer Solutions to Linear Programs," Bulletin of the American Mathematical Society, Vol. 64, 1958, pp. 275-278.

20. Graves, J., "On the Storage and Handling of Binary Data Using FORTRAN with Applications to Integer Programming," Operations Research, Vol. 27, No. 3, 1979, pp. 534-547.

21. Held, M. and R. Karp, "A Dynamic Programming Approach to Sequencing Problems," Journal of the Society of Industrial and Applied Mathematics, Vol. 10, No. 1, 1962, pp. 196-210.

22. Held, M. and R. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees," Operations Research, Vol. 18, No. 6, 1970, pp. 1138-1162.

23. Held, M. and R. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees: Part II," Mathematical Programming, Vol. 1, No. 1, 1971, pp. 6-25.

24. Lin, S. and W. Kernighan, "An Effective Heuristic Algorithm for the Traveling-Salesman Problem," Operations Research, Vol. 21, No. 2, 1973, pp. 498-516.

25. Miliotis, P. "Integer Programming Approaches to the Travelling Salesman Problem," Mathematical Programming, Vol. 10, 1976, pp. 367-378.

26. Miller, C., A. Tucker and R. Zemlin, "Integer Programming Formulation of Traveling Salesman Problems," JACM, Vol. 7, 1960, pp. 326-329.

27. Norback, J. and R. Love, "Geometric Approaches to Solving the Traveling Salesman Problem," Management Science, Vol. 23, No. 11, 1977, p. 1208.

28. O'Neil, B. and D. Whybark, "The Multiple-Vehicle Routing Problem," The Logistics and Transportation Review, Vol. 11, No. 2, 1975, pp. 153-163.

29. O'Neil, B. and D. Whybark, "Vehicle Routing from Central Facilities," International Journal of Physical Distribution, Vol. 2, No. 1, 1972, pp. 93-97.

30. Orloff, C., "A Fundamental Problem in Vehicle Routing," Networks, Vol. 4, No. 1, 1974, pp. 35-64.

31. Orloff, C., "Routing a Fleet of M Vehicles to/from a Central Facility," Networks, Vol. 4, No. 2, 1974, pp. 147-162.

32. Papadimitriou, C. and K. Steiglitz, "Some Examples of Difficult Traveling Salesman Problems," Operations Research, Vol. 26, No. 3, 1978, p. 434.

33. Picard, and Queyranne, "Traveling Salesman and Scheduling Problems," Operations Research, Vol. 26, No. 1, 1978, pp. 86-110.

34. Psaraftis, H., "A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem," Transportation Science, Vol. 14, No. 2, May, 1980, pp. 130-154.

35. Rosenkrantz, D., R. Stearns and P. Lewis, "An Analysis of Several Heuristics for the Traveling Salesman Problem," SIAM Journal on Computing, Vol. 6, No. 3, 1977, p. 563.

36. Russell, R. A., "An Effective Heuristic for the M-Tour Traveling Salesman Problem with Some Side Constraints," Operations Research, Vol. 25, No. 3, 1977, pp. 517-524.

37. Stein, David M., "Scheduling Dial-a-Ride Transportation Systems," Transportation Sciences, Vol. 12, No. 3, Aug 1978, pp. 232-249.

38. Tillman, F. and T. Cain, "An Upper Bounding Algorithm for the Single and Multiple Terminal Delivery Problem," Management Science, Vol. 18, No. 11, 1972, pp. 664-682.

39. Tillman, F. and H. Cochran, "A Heuristic Approach for Solving the Delivery Problem," The Journal of Industrial Engineering, Vol. 19, No. 7, 1968, pp. 354-358.

40. Swestka, J. and V. Huckfeldt, "Computational Experience with a M-Salesman Traveling Salesman Algorithm," Management Science, Vol. 19, No. 7, 1973, pp. 790-799.

41. Webb, M., "Some Methods of Producing Approximate Solutions to Travelling Salesman Problems with Hundreds or Thousands of Cities," Operational Research Quarterly, Vol. 22, No. 1, 1971, pp. 49-66.

42. Wren, A. and A. Holliday, "Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points," Operational Research Quarterly, Vol. 23, No. 3, 1972, pp. 333-344.

43. Yellow, P., "A Computational Modification to the Savings Method of Vehicle Scheduling," Operational Research Quarterly, Vol. 21, No. 3, 1970, p. 281.

VITA

Gerald R. Armstrong was born in Peoria, Illinois, on February 16, 1944. After attending schools in Cincinnati, Ohio, and Springfield, Illinois, he was graduated from Peoria High School in June 1962. Four years later in June 1966 he received a Bachelor of Arts degree in Chemistry from Bradley University.

Upon graduation he entered the United States Air Force and progressed to his current rank of Major. He earned a Master of Science degree in Logistics Management in February 1971 from the Air Force Institute of Technology (AFIT). In September 1977 he entered the Graduate School of The University of Tennessee, Knoxville, under AFIT sponsorship. He received the Doctor of Philosophy degree in Management Science in June 1981.

The author is currently on AFIT's Graduate Faculty, Operational Sciences Department, School of Engineering, Wright-Patterson AFB, Ohio.

He is married to the former Toni J. Saccenti of Peoria Heights, Illinois. They have three children, Renee, Kimberlee, and Timothy.